GRAN ENCICLOPEDIA INFORMATICA

INFORMATICA BASICA/ 2

EDICIONES NUEVA LENTE

GRAN ENCICLOPEDIA INFORMATICA

EDICIONES NUEVA LENTE


INFORMATICA
BASICA
FROM: F. Lara
TO: Everybody
Nueva Lente
apricot F1

SUMARIO

Una publicación:
Ediciones Nueva Lente, S. A.

Director editor: MIGUEL J. GOÑI
Director de producción: SANTOS ROBLES.

Director de la obra: FRANCISCO LARA.

Colaboradores: PL/3 - MANUEL MUÑOZ - ANGEL MARTINEZ - MIGUEL DE ROSENDO - DAVID SANTAOLALLA - SANTIAGO RUIZ - LUIS COCA - MIGUEL ANGEL VILA - MIGUEL ANGEL SANCHEZ-VICENTE ROBLES.

Diseño: BRAVO/LOFISH.

Maquetación: JUAN JOSE DIAZ SANCHEZ.

Ilustración: JOSE OCHOA.

Fotografía: (Equipo Gálata) ALBINO LOPEZ y EDUARDO AGUDELO.

Ediciones Nueva Lente, S. A.:

Dirección y Administración:
Benito Castro, 12. 28028 Madrid.
Tel.: 245 45 98.

Números atrasados y suscripciones:
Ediciones Ingelek, S. A.
Plaza de la Rep. Ecuador, 2 - 1.°. 28016 Madrid.
Tel.: 250 58 20.

Plan general de la obra:
18 tomos monográficos de aparición quincenal.

Distribución en España:
COEDIS, S. A. Valencia, 245. Tel.: 215 70 97.
08007 Barcelona.

Delegación en Madrid:
Serrano, 165. Tel.: 411 11 48.

Distribución en Argentina:
Capital: AYERBE
Interior: DGP

Distribución en Chile:
Alfa Ltda.

Distribución en México:
INTERMEX, S. A.
Lucio Blanco, 435
México D.F.

Distribución en Uruguay:
Ledian, S. A.

Edita para Chile:
PYESA
Doctor Barros Borgoño, 123
Santiago de Chile

Importador exclusivo Cono Sur:
CADE, SRL. Pasaje Sud América, 1532.
Tel.: 21 24 64.
Buenos Aires - 1.290. Argentina.

Fotomecánica: Ochoa, S. A.
Miguel Yuste, 32. 28037 Madrid.

Impresión: Gráficas Reunidas, S. A.
Avda. de Aragón, 56. 28027 Madrid.

ISBN de la obra: 84-7534-184-5.
ISBN del tomo 2: 84-7534-186-1.

Printed in Spain
Depósito legal: M. 27.605-1986

Precio de venta al público en Canarias, Ceuta y Melilla: 940 ptas.
Septiembre 1986.

Dialogando con el ordenador

Los elementos del software



l ordenador electrónico, por sí solo, es una máquina esencialmente inútil, incapaz de realizar tarea alguna. Todo su secreto reside en la capacidad de recibir las órdenes del usuario, interpretarlas y ejecutarlas con disciplina y absoluta eficacia.

En resumidas cuentas, el hardware o conjunto de elementos físicos deben recibir una detallada y completa «educación» para que puedan prestar cualquier tipo de servicio al usuario.

Cabe equiparar a un sistema informático con un triángulo equilátero cuyos vértices están ocupados por el «hardware» o equipo físico, el «software» o conjunto de órdenes que instruyen al hardware, y el hombre. Sin lugar a dudas, el personal informático ocupa el vértice superior de la referida pirámide; a fin de cuentas, es el hombre quien diseña la máquina, la instruye y decide el camino a seguir.

¿Qué es el software?

La característica esencial de los ordenadores reside en su naturaleza de máquinas programables y dotadas de una capacidad para memorizar información. Y no hay que perder de vista que el objetivo de los ordenadores es, precisamente, el tratamiento de la información.

En su sentido más amplio, el término «software» identifica a todo cuanto complementa a la arquitectura física del ordenador y lo convierte en una máquina «instruible», versátil y dispuesta a ejecutar innumerables tareas distintas.

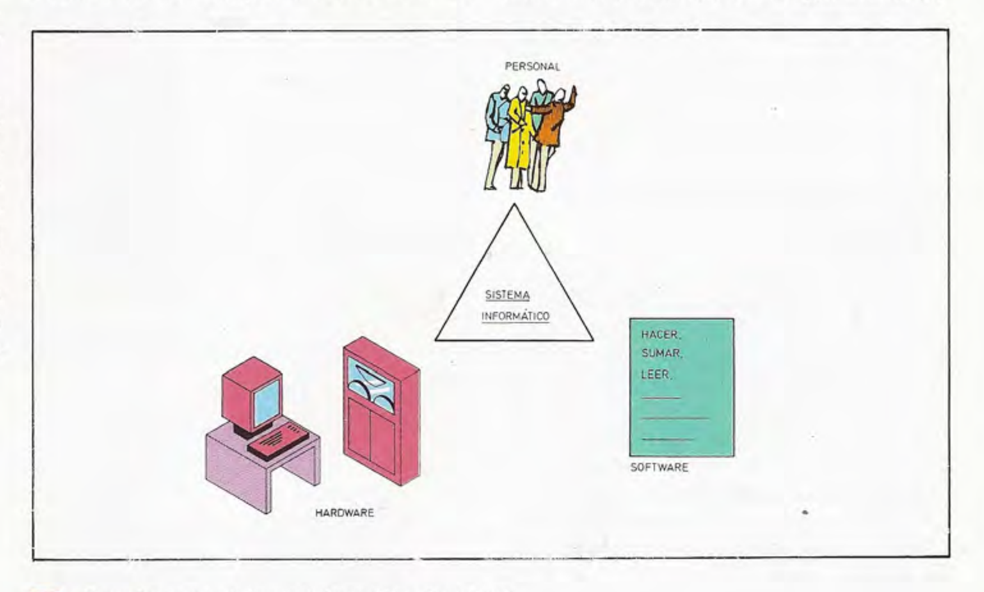
Adoptando una acepción más estricta, habría que hablar del software como del conjunto de órdenes, instrucciones y programas que «dan vida» al hardware del ordenador: el denominado componente lógico del ordenador.

Forman parte del software tanto los programas que instruyen a la máquina para mecanizar la gestión contable, apoyar la edición de texto escrito, realizar complicadas operaciones, confeccionar la nómina de una empresa... como los programas internos que se ocupan de que el hardware acepte órdenes introducidas por el teclado, visualice los datos a través de una pantalla y enseñe a los circuitos del ordenador a reconocer e interpretar las instrucciones de los programas de aplicación y a ejecutarlas oportunamente.

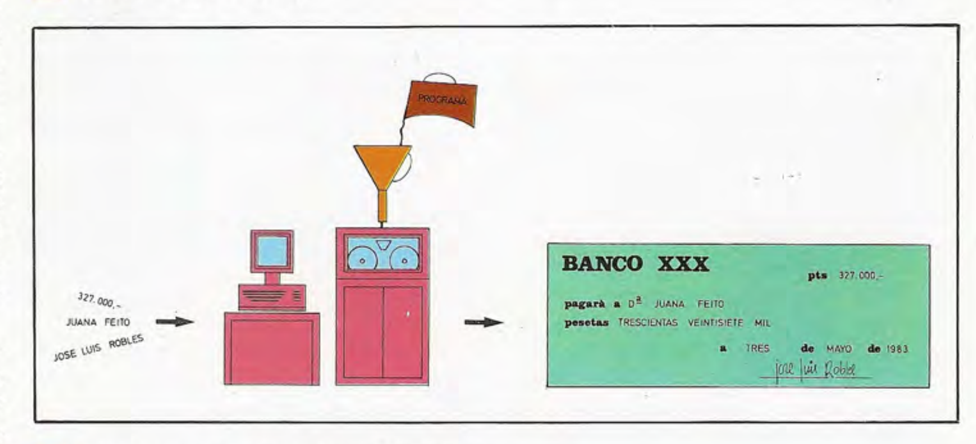
Para que un ordenador curse una determinada tarea es preciso introducir en la máquina una secuencia organizada de instrucciones —el denominado programa— que, ejecutadas una tras otra, concluyan con la resolución de la referida tarea.

Pero, claro, para que el ordenador pueda leer el programa de aplicación del usuario, debe haber sido instruido previamente para ello. De esta misión se ocupará, por ejemplo, un programa inicial de lectura.

La forma en la que ese programa inicial se introduce en el ordenador ha variado con los tiempos. Antiguamente se



Para que un sistema informático funcione hace falta la participación de tres elementos: el material (hardware), el lógico (software) y el humano (personal).



El programa es el responsable de que el sistema ordenador pueda transformar los datos de entrada en una información resultante de salida.

hacía a mano, a través del teclado de la consola; unas veces en forma de secuencia de códigos binarios, otras en forma decimal. Posteriormente se introdujo un *dispositivo de autocarga;* esto es: el programa inicial se encuentra en una memoria permanente y entra en actividad mediante la simple presión de una tecla. Este programa inicial permitía que el ordenador leyera un programa de lectura más elaborado, programa que residía en un disco o una cinta y que estaba escrito en el lenguaje propio del ordenador. Por último, este programa permitía ya la lectura del programa de trabajo.

Nacen los lenguajes de programación

Con el fin de facilitar el trabajo del programador, surge la necesidad de que el ordenador entienda un lenguaje diferente al suyo propio. Entramos en la etapa de la *simbolización.* Ya el programador no necesita conocer realmente dónde ubica sus datos, le basta con referirse a direcciones simbólicas. Así nacen los lenguajes de programación del tipo ensamblador y, consecuentemente, nace el *software traductor* o conjunto de programas que permiten convertir los programas escritos en el lenguaje del programador al lenguaje que entiende la máquina.

Por esta vía se avanza más y se llega a un nuevo paso que permite al hombre dar al ordenador las fórmulas o notaciones que normalmente usa en su trabajo. Aparecen los lenguajes de programación de alto nivel y los *compiladores:* programas cuya misión es traducirlos al lenguaje del ordenador.

La aparición de los sistemas operativos

El avance tecnológico de los ordenadores incrementa su capacidad de trabajo, con lo que el hombre tiene muchas más dificultades en aprovecharlo y, por tanto, necesita nuevas ayudas en forma de programas que le faciliten no sólo la programación, sino también la explotación de los equipos.

Aparecen programas que facilitan los cálculos corrientes, programas que permiten la transferencia entre sorportes de memoria y programas que reducen los tiempos muertos de la máquina. La organización de los trabajos deja de ser confiada al hombre para ser controlada por un software específico: *el sistema operativo.*

El sistema operativo es un programa (o más propiamente un conjunto de programas) que permite encadenar trabajos, simultanear diversos elementos periféricos, conectar adecuadamente los periféricos, proporcionar protección contra los errores, contabilizar los tiempos de utilización de las diversas unidades, etc.

PORCENTAJE DEL COSTO
100, 80, 60, 40, 20
HARDWARE
DESARROLLO DEL SOFTWARE
MANTENIMIENTO DEL SOFTWARE
1955 1970 1985
AÑO
FUENTE: RELIABILITY, MANAGEMENT, METHODS AND MATHEMATICS
POR: LLOYD & LIPOW

El elemento lógico —el software— ha ido incrementando su participación en el coste total del sistema.



En sus primeras etapas de evolución, los ordenadores sólo podían procesar los programas uno tras otro. Hasta que no concluía uno no podía empezar la ejecución del siguiente.



Una de las funciones del sistema operativo es la de controlar las conexiones del ordenador con los periféricos asociados al mismo.

El siguiente paso para mejorar las relaciones entre el hombre y la máquina se encuentra en aprovechar los tiempos muertos de la máquina para ejecutar otros programas. Aparece la *multiprogramación.* El sistema operativo se complica con el fin de permitirle elegir en cada momento el programa que debe ejecutar, en función de los elementos del ordenador no utilizados en ese instante.

El siguiente paso consiste en conectar varios ordenadores, con lo que el sistema operativo pasa a controlar el *multiproceso.*

La nueva mejora en el rendimiento de la explotación de los equipos aparece con la explotación en *tiempo real,* gracias a la cual el usuario puede obtener respuesta inmediata a su problema.

La explotación en tiempo real permite al usuario disponer de todo el ordenador, pero durante la ejecución de su programa hay pérdidas de tiempo del procesador. Por consiguiente, el sistema operativo se perfecciona y llega a la explotación en *tiempo compartido* de un solo ordenador por varios usuarios.

Tipos de software

Podemos clasificar a los programas en cuatro grandes grupos:

- *Software específico* o programas para procesar datos. Constituido por los programas de aplicación y que pueden ser escritos tanto por el constructor como por el usuario u oficinas de consulting. Todos los programas que resuelven los problemas de gestión, técnicos, científicos, etc., pertenecen a este grupo.

- *Software traductor* o programas de ayuda para escribir nuevos programas. Constituido por los programas que permiten que los programas escritos por los usuarios en un lenguaje distinto al de la máquina se conviertan en programas con instrucciones en código de máquina. Son escritos y proporcionados por los fabricantes.

- *Software funcional* o programas de ayuda para ejecutar otros programas. Más comúnmente conocido como sistema operativo. Es un conjunto de programas que facilitan una explotación más racional de los ordenadores, guiando todas las tareas y ayudando a los programas en ciertas funciones. Son desarrollados por los constructores.

- *Software general* o rutinas de utilidad. Programas que permiten la realización de funciones de uso frecuente y que generalmente son escritos por los fabricantes, aunque también pueden desarrollarlos los propios usuarios.

En cualquier caso y sea cual fuere su catalogación, el software es el componente lógico que actúa sobre el hardware o circuitería física, y permite que el ordenador pueda realizar su trabajo.

El software es, por consiguiente, el conjunto de programas que controlan el funcionamiento del ordenador. También sabemos que los programas están formados por instrucciones, que son los elementos básicos y esenciales del software.

Tipos de instrucciones

Un programa, como sabemos, es una secuencia ordenada de instrucciones. Estudiar todas las instrucciones que se utilizan en el mundo del ordenador es tarea prácticamente imposible, ya que cada ordenador está fabricado para manejar un cierto número de ellas y cada lenguaje de programación tiene las suyas propias. Pensemos en lo distintas que pueden ser las instrucciones de un ordenador científico de las de otro destinado a aplicaciones administrativas. Independientemente del lenguaje de programación, podemos clasificar las instrucciones en:

a) **Instrucciones de comienzo/parada**

Estas instrucciones señalan el comienzo o la detención de un programa. Un programa se puede «parar» por varios motivos:

1. Porque el programa haya concluido.
2. Porque exista un error en alguna de las instrucciones.
3. Porque sea necesaria la intervención del operador.

b) **Instrucciones de cálculo aritmético**

Son aquellas que efectúan el cálculo de las operaciones aritméticas: suma, resta, multiplicación y división.

c) **Instrucciones de cálculo lógico**

Las que realizan las operaciones booleanas y de decisión, basadas en variables que pueden tomar los valores «verdadero» (TRUE) y «falso» (FALSE).

d) **Instrucciones de transferencia de control**

Son las que rompen la ejecución secuencial de las instrucciones del programa, normalmente como resultado de verificar si cumple alguna condición aritmética o lógica. A estas instrucciones también se les llama de «SALTO» y van asociadas siempre a una toma de decisión.

e) **Instrucciones de entrada/salida**

Realizan la comunicación entre la unidad central del ordenador y los elementos de la periferia.

f) **Instrucciones de definición**

Definen las constantes, formatos, zonas de reserva de memoria, etc. Con estas instrucciones definimos, por ejemplo, cuál es el tamaño de una matriz de datos.

g) **Instrucciones modificadoras de instrucciones**

Permiten modificar códigos de operación o direcciones con el fin de que el programa se corrija a sí mismo, permitiendo un ahorro de posiciones de memoria.

h) **Instrucciones de transferencia de datos**

Son las que permiten el intercambio o copia de información de una zona a otra de memoria.

i) **Instrucciones de edición**

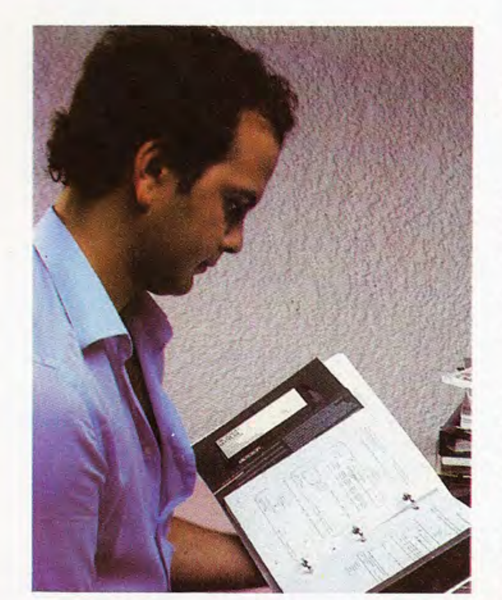
Facilitan la programación de las entradas y salidas.

j) **Instrucciones de conversión de formatos**

Cambian los formatos en que la información se almacena.



En los actuales sistemas con multiprogramación, el ordenador puede aprovechar los tiempos muertos en un programa para ocuparse de la ejecución de otro. Su velocidad de proceso le permite distribuir su tiempo atendiendo a varios usuarios.



La adeucada combinación de los elementos del software proporcionará al usuario los medios oportunos para la correcta y eficaz explotación de los sistemas informáticos.

Instrucciones

Comúnmente se entiende por instrucciones «el conjunto de reglas o normas dadas para la realización o empleo de algo».

En informática, *instrucción* es la información que comunica a un ordenador una acción elemental a ejecutar.

Recordemos que una orden aislada no permite realizar el proceso completo, sino que es necesario un conjunto de instrucciones colocadas en un orden secuencial lógico.

Por ejemplo, si queremos elaborar una tortilla de patatas, tendremos que ejecutar una serie de instrucciones: pelar las patatas, batir los huevos, freír las patatas, etc.

Es evidente que estas instrucciones tienen que ejecutarse en un orden adecuado, ¡no se van a pelar las patatas después de freírlas!

Así pues, una instrucción por separado no nos dice mucho; para obtener el resultado necesitamos ejecutar el conjunto de todas las instrucciones debidamente ordenadas.

Algoritmo

En el ejemplo anterior hemos visto que para conseguir la tortilla hay que seguir una serie de pasos detalladamente especificados. Esto nos ayuda a aclarar la ardua definición de algoritmo: serie de instrucciones, en una cierta secuencia, necesarias para describir las operaciones que llevan a la resolución de un problema.

Programa

Un programa es una serie de instrucciones, perfectamente legibles por el ordenador y destinadas a realizar un determinado trabajo o solucionar un problema. Esta definición es similar a la de algoritmo, con la diferencia de que el programa, en vez de utilizar un lenguaje humano, emplea un lenguaje inteligible por la máquina.

El programa es adecuado para una relación hombre-máquina, mientras que el algoritmo lo es para una relación hombre-hombre.

Si quisiéramos calcular una suma de dos números utilizando un ordenador, tendríamos que darle las siguientes instrucciones:

- Leer los datos (cantidades a sumar) del dispositivo de entrada (por ejemplo, la tarjeta perforada) y almacenarlos en la memoria.

- Sumar los dos números.

• Almacenar el resultado de la suma en la memoria.

• Mostrar el resultado en la pantalla. Estas instrucciones dadas al ordenador, en este orden, constituyen el *programa* para la suma de dos números. No se puede alterar el orden de las instrucciones, ya que si se hiciera no obtendríamos el resultado deseado.

Estas instrucciones, que forman el programa, se almacenan internamente en el ordenador. Una vez almacenadas se activará su ejecución introduciendo una instrucción de comienzo de programa: el resultado será la ejecución secuencial de las sucesivas instrucciones y la obtención del valor resultante de la suma de los dos números.

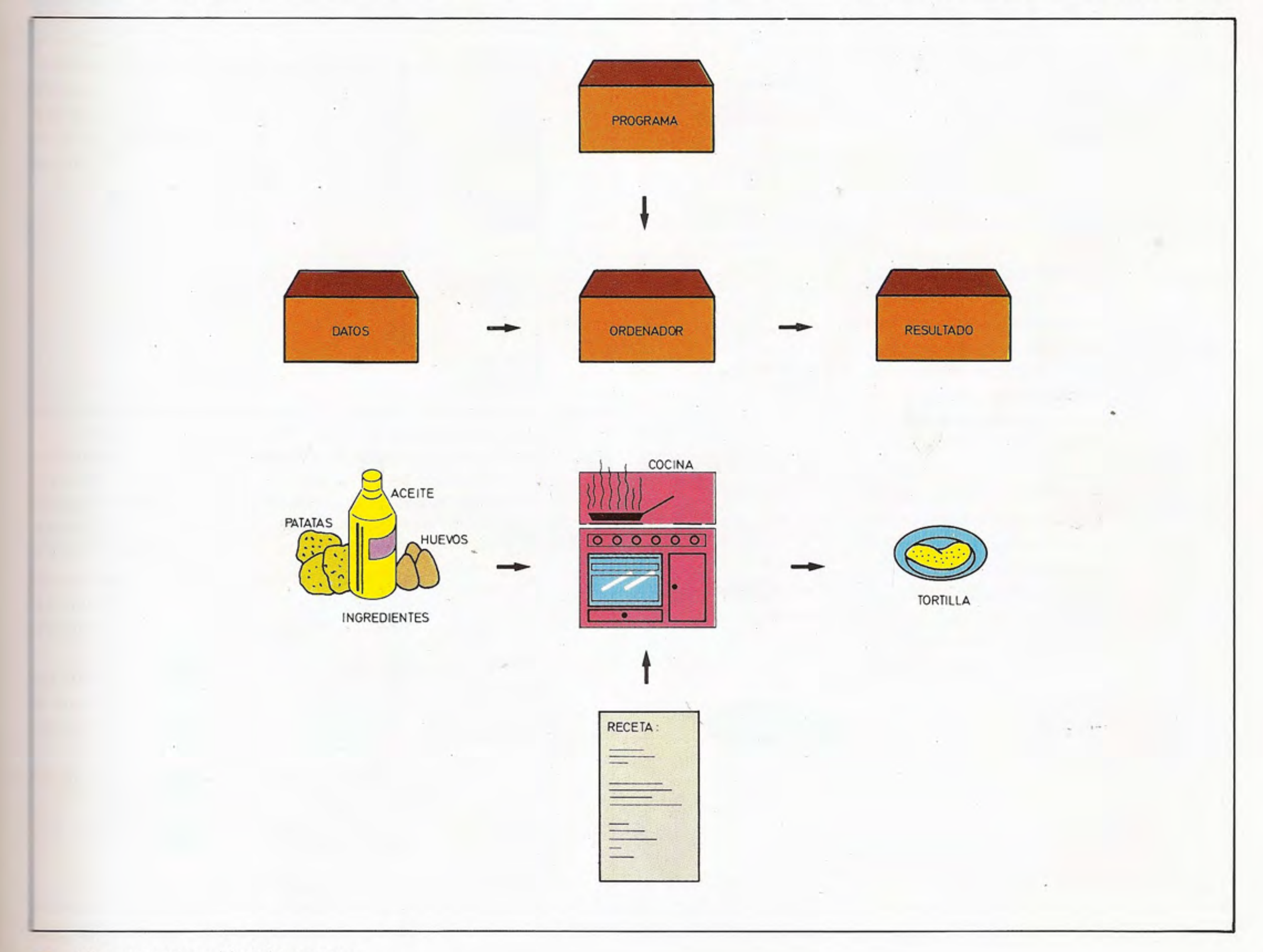
Tipos de programas

Existen diversos tipos de programas, siendo los principales los que se relacionan a continuación:

1. *Programas lineales*

El desarrollo de la ejecución del programa se realiza en el mismo orden secuencial en el que se han escrito las instrucciones.

Un ejemplo de este tipo de programas es una cadena de fabricación de coches. El coche entra en la cadena con sólo el chasis; pasa por el puesto de colocación del motor; a continuación se le colocan las ruedas. Una vez colocadas las ruedas, la cadena lo lleva al puesto de instalación eléctrica, faros, intermitentes, luces de posición, etc. Completada esta



Un programa, al igual que una receta de cocina, es el conjunto de instrucciones que permiten pasar de los datos (ingredientes) a la información resultante (plato).

operación, la cadena lo pasa a la sección de montaje de cristales para emplazar el parabrisas, ventanillas laterales, guardabarros, etc. Seguidamente, la cadena lleva al coche a la zona de pintura y una vez pintado sale de la cadena de fabricación. Es un programa secuencial: el coche una vez dentro de la cadena no puede ser vuelto atrás, sino que debe completarla pasando por todas las secciones de la misma.

2. *Programas cíclicos*

Son programas que contienen un grupo de instrucciones que se van a repetir un cierto número de veces y, por consiguiente, tienen que contener instrucciones de bifurcación o transferencia de control.

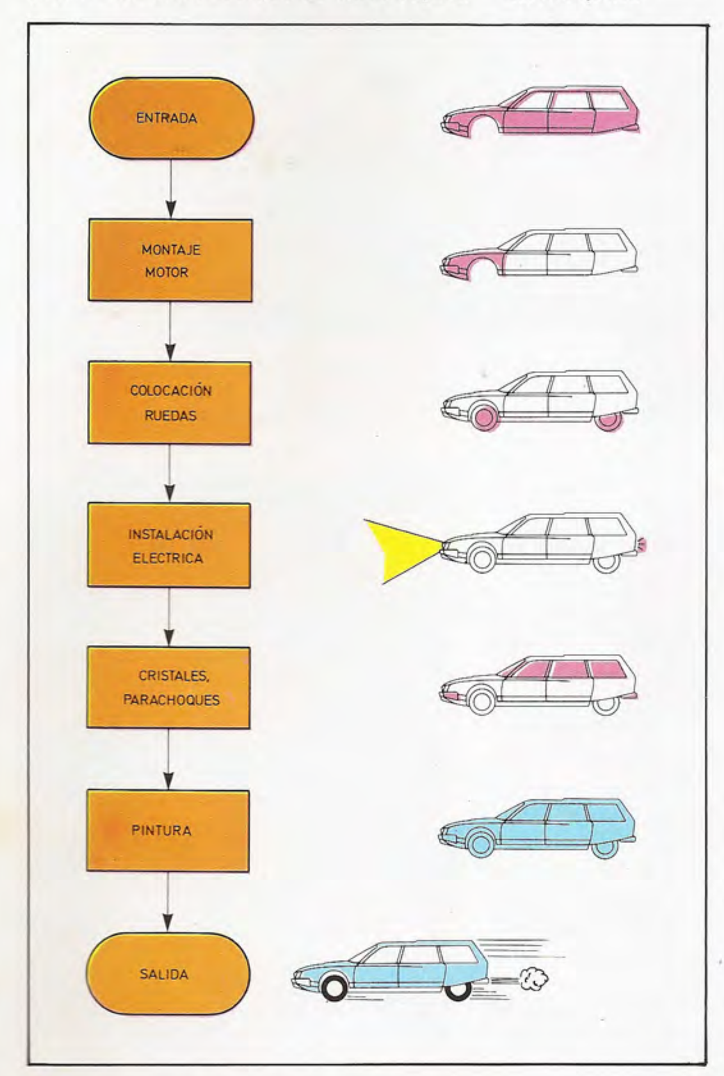
Un ejemplo de este tipo de programas puede ser el de un «scalextric» en el que queremos que un coche dé 50 vueltas.

Colocamos el coche en la pista y el contador de vueltas a cero. Cuando el coche da una vuelta, el contador pasa a 1 e inicia la segunda vuelta. Al completarla el contador pasa a 2, y así sucesivamente hasta que este último llegue a 50. En ese preciso instante el coche se para.

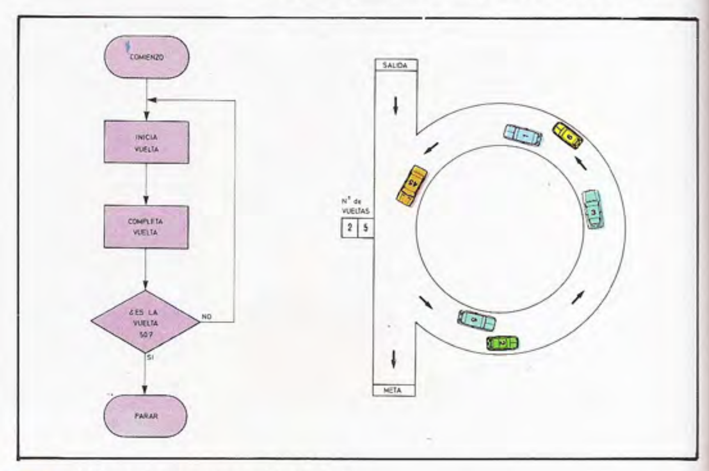
3. *Programas alternativos*

Son los que pueden continuar por diversos caminos según los valores que tomen ciertas variables, bien en la entrada de datos o en cualquier momento de la ejecución.

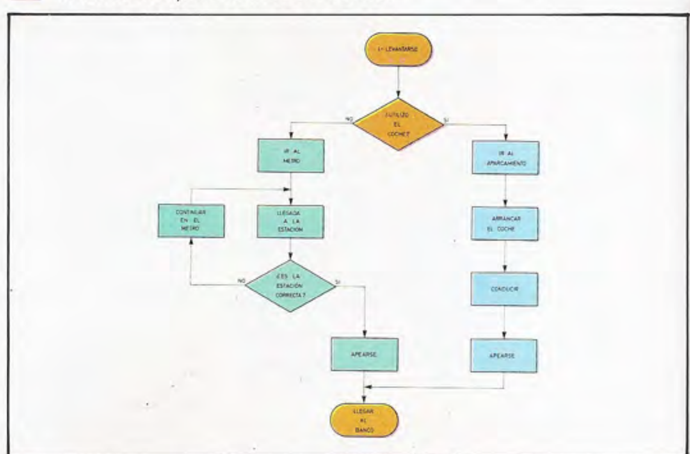
Vamos a explicar esta definición por medio de un ejemplo. Cuando una persona sale de casa para ir a trabajar a un banco puede utilizar como transporte su coche o ir en metro. Si utiliza su coche, tiene que ir al aparcamiento, arrancarlo e ir hacia la oficina. Finalmente aparca y entra en el banco. Si no utiliza coche, tiene que tomar el metro e ir observando



La secuencia de desarrollo de un proceso lineal guarda un total paralelismo con la secuencia de procesos que tienen lugar en una cadena de montaje.



Al igual que sucede en una competición automovilista, en un programa cíclico se repite un determinado número de veces el mismo bloque de instrucciones.



En un programa alternativo puede elegirse entre diversos caminos que conducen a un mismo destino o resultado.

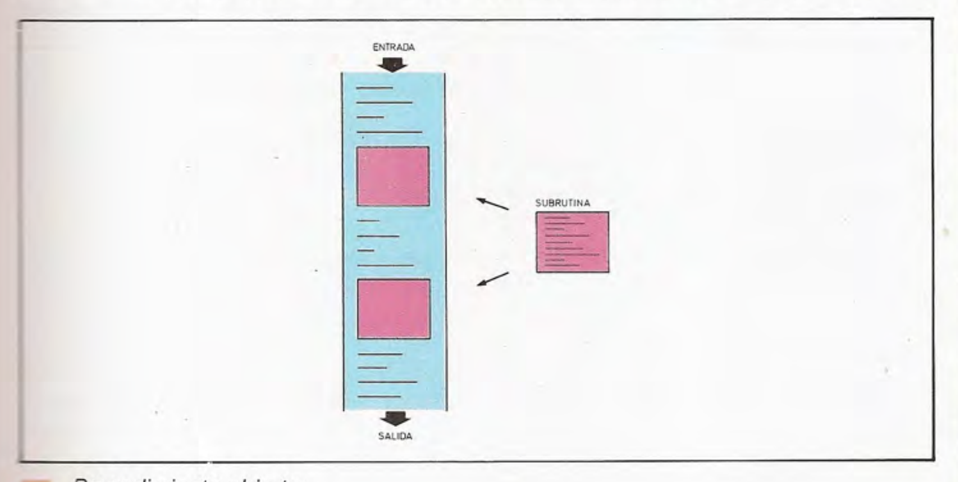
las estaciones. Si el metro llega a una estación que no es la del empleado, éste no se apeará. Cuando llegue a la suya se apeará y entrará en el banco.

Rutina

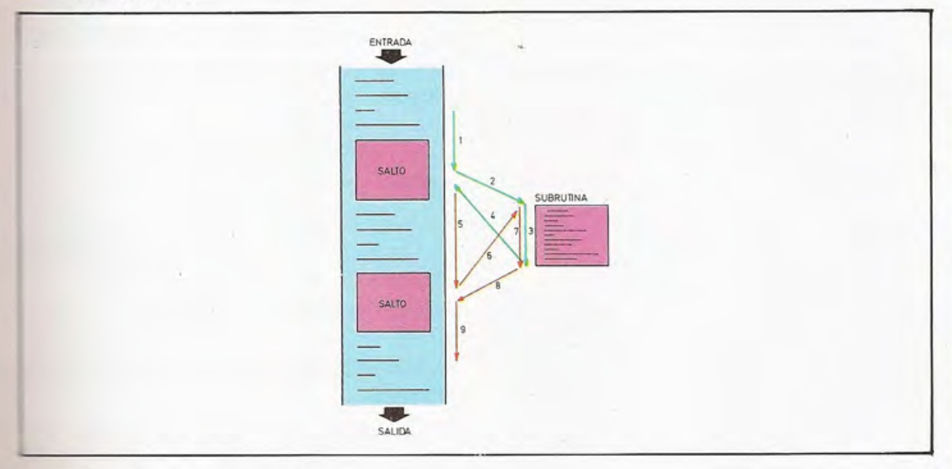
Es un conjunto de instrucciones que cumple un cometido concreto en un programa y que normalmente sólo se ejecuta una vez. Por ejemplo, el cálculo de los costes de la Seguridad Social de los empleados de una empresa.

Subrutina

Existen programas que contienen un conjunto de instrucciones que pueden intervenir varias veces en la ejecución del mismo. Suponga que tenemos un programa en el que existen una serie de instrucciones que se repiten en diversas zonas del mismo. Podríamos sacar del programa esas instrucciones que se repiten y formar con ellas una *subrutina.*



Procedimiento abierto: la subrutina se intercala en el programa cuando se necesita su intervención.



Procedimiento cerrado: la secuencia de instrucciones «salta» para ejecutar la subrutina.

Desarrollo histórico del software

El software se ha desarrollado paralelamente al avance de los ordenadores, adquiriendo características especiales con cada generación. Para los ordenadores de la primera generación, los programas debían escribirse en código de máquina, con todas las dificultades que esto implicaba, ya que no sólo había que reconocer los códigos de las instrucciones, sino también las direcciones absolutas de memoria de todos los datos. Por ello, aparecen muy rápidamente los primeros lenguajes simbólicos que, en general, eran nemotécnicos. El uso de estos lenguajes obliga a decodificarlos mediante programas traductores o ensambladores, con lo que se inicia el *software traductor.*
También en esta época los constructores proporcionan pequeñas bibliotecas de rutinas, con lo que aparece el software de *rutinas de utilidad.*
Con la segunda generación de ordenadores aparecen los Autocoder (lenguajes ensambladores básicos) y el Cobol (lenguaje destinado a la gestión), junto con gran cantidad de rutinas. Y, naturalmente, surgen los compiladores. Muchas de las rutinas son útiles a los usuarios para las operaciones más elementales (copias de información, clasificación de ficheros, etc.).
También nacen los sistemas de explotación, que son programas que permiten el encadenamiento rápido de los programas del usuario. Estos primeros *sistemas operativos* sólo se utilizaban en los equipos más grandes.
En la tercera generación ya hay sistemas operativos para todos los ordenadores y se diseñan nuevos lenguajes.

La definición de subrutina es: conjunto de instrucciones que se pueden ejecutar un número ilimitado de veces. Las subrutinas pueden ser llamadas por un solo programa o bien por otros programas que se encuentren en la memoria del ordenador.

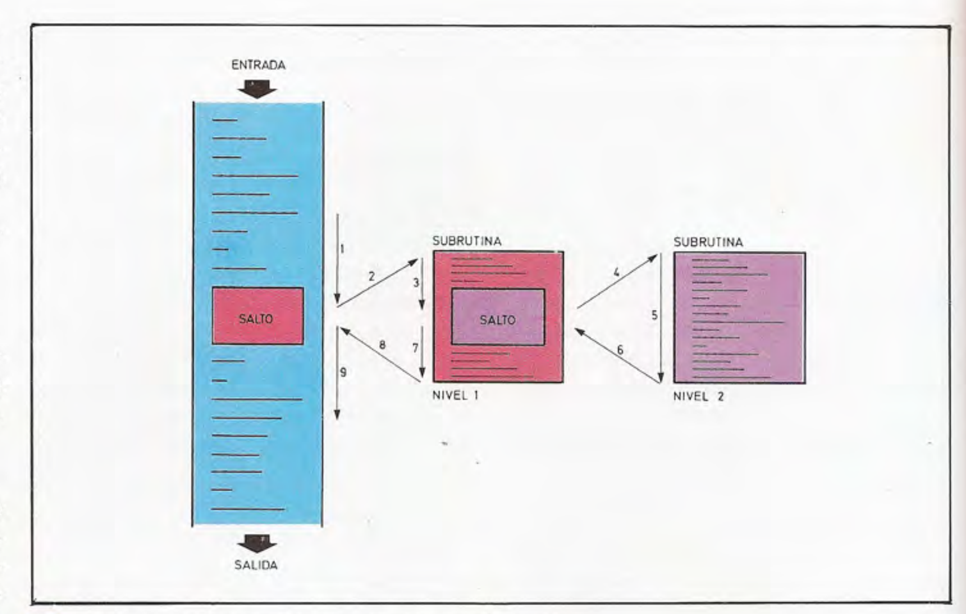
Para incorporar la subrutina al programa se pueden seguir dos caminos:

1. *Procedimiento abierto*

Intercalando en el programa la subrutina cada vez que se necesite, con lo que no se reduce el espacio de memoria.

2. *Procedimiento cerrado*

Cuando hay que utilizar la subrutina se efectúa un salto al comienzo de la misma. Esta, a su vez, termina con una instrucción de retorno al programa de partida. Existen también las subrutinas anidadas que son, a su vez, subrutinas de otras subrutinas.



El número de orden que acompaña a la flecha define la secuencia en la que se ejecutan las instrucciones de un «anidamiento de subrutinas».

Para saber más

¿Es necesario el ordenador para procesar datos?

No. La informática es la ciencia del tratamiento automatizado de la información. Tratar datos se ha hecho siempre; la diferencia es que hoy en día es más práctico utilizar ordenadores que recurrir al clásico proceso manual.
En realidad el proceso no cambia, sólo que se hace más rápidamente con el ordenador.

¿Qué significa software?

Es un término inglés formado por las palabras SOFT, que significa «blando», y WARE, que en castellano no tiene un significado concreto.
El término nació en contraposición a HARDWARE (hard significa duro), denominación aplicada a los elementos físicos que constituyen el ordenador.

¿Se puede llamar al software con otros nombres?

Aunque no están muy extendidos, en publicaciones en lengua castellana se pueden también encontrar los términos logicial o logical, que significa «lógico, lo no material». Procede, al igual que la palabra «informática», de la terminología francesa.

¿Es caro el software?

El software es realizado por personas y, por lo tanto, su costo está en función del salario de analistas y programadores. Por otra parte, el avance tecnológico ha reducido los costes del hardware. Y lo que ha sucedido es que el porcentaje de coste de desarrollo del software se ha incrementado en los últimos años respecto al coste total del sistema informático.

¿Cuándo un programa no tendría solución?

Cuando el ordenador llegara a una instrucción que, por fallo del programador o error en los datos, no la pudiera ejecutar. Esta situación puede llegar a originar la parada del ordenador, a no ser que se tomen las debidas precauciones.

¿Cuántos ciclos puede tener un programa?

Un programa debe tener un número finito de ciclos, pues de lo contrario el ordenador no se pararía nunca.

¿Cómo controla el ordenador el salto a la subrutina?

El ordenador traslada a la subrutina los datos que ésta necesita para realizar sus operaciones y «recuerda» el lugar donde debe continuar el programa cuando acabe la subrutina.

¿Cuántos niveles de anidamiento puede tener un programa?

En teoría, no hay inconveniente en que una subrutina llame, sucesivamente, a otras subrutinas. En la práctica dependerá de las características de diseño del ordenador, que siempre limitan el número de retornos que puede «recordar».

Los lenguajes informáticos

Máquinas poliglotas



Para que el ordenador pueda llevar a cabo los procesos que desee el usuario, es necesario proporcionarle el adecuado conjunto de instrucciones agrupadas y ordenadas en lo que se denomina *programa.*

El procesador irá extrayendo las instrucciones de la memoria central con el fin de proceder a su ejecución. Por razones tecnológicas, la memoria sólo almacena dígitos binarios (bits: ceros o unos); por tanto, las únicas instrucciones que el ordenador es capaz de entender son combinaciones de unos y ceros: instrucciones elaboradas en *código de máquina.*

Las instrucciones en código máquina son difícilmente comprensibles a primera vista, aun cuando en lugar de representarlas en binario se escriban en notación hexadecimal. Por ello, la elaboración de un programa se convierte en una tarea dura y sujeta a muchos errores. Por otra parte, aparece la dificultad adicional de que cada ordenador tiene su propio juego de instrucciones elementales.

Inconvenientes del lenguaje máquina

En teoría, dado que el ordenador debe operar con instrucciones que le sean comprensibles, es condición necesaria que reciba una programación en lenguaje de máquina. No obstante, este tipo de programación presenta tres graves inconvenientes:

a) El programador debe conocer del orden de un centenar de instrucciones elementales, además de asignar a cada instrucción, dato, variable y resultado una dirección real de memoria y recordar, durante la programación, la dirección asignada. (De alguna forma deberá llevar un plano de la memoria.)

Como quiera que un programa pequeño puede alcanzar fácilmente el centenar de instrucciones, las dificultades aumentan a medida que crece el tamaño del programa.

b) Las instrucciones de nivel máquina sólo contemplan la programación de operaciones elementales. Por tanto, el programador debe conocer muy a fondo la estructura del ordenador que utiliza y descomponer los procesos a resolver en operaciones elementales que formen parte del repertorio del ordenador.

c) Quizá la dificultad más grave es que después del gran esfuerzo realizado para hacer un trabajo en estas condiciones, el resultado —el programa en código de máquina— sólo puede ejecutarse en un tipo de ordenador, ya que distintos ordenadores hablan diferentes lenguajes máquina.

Para eliminar estos inconvenientes se crearon lenguajes de programación cada vez más alejados del lenguaje de la máquina y más próximos al lenguaje humano.

Los sucesivos niveles de los lenguajes de programación, cada vez más evolucionados, permiten ir eliminando los inconvenientes citados.

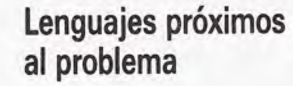
Lenguajes de ensamble

Este tipo de lenguaje sortea el primero de los inconvenientes enunciados en el párrafo anterior. Utiliza códigos nemotécnicos en lugar de códigos binarios y direcciones simbólicas de memoria en lugar de direcciones absolutas. Los símbolos de los códigos de operación son fijos para cada lenguaje y las direcciones simbólicas las puede elegir el programador respetando unas ciertas reglas.

En este tipo de lenguajes, denominados «de ensamble» o ensambladores, las instrucciones siguen manteniendo una cerrada similitud con las instrucciones elementales del código máquina, por lo que el programador necesita seguir conociendo a fondo su ordenador.

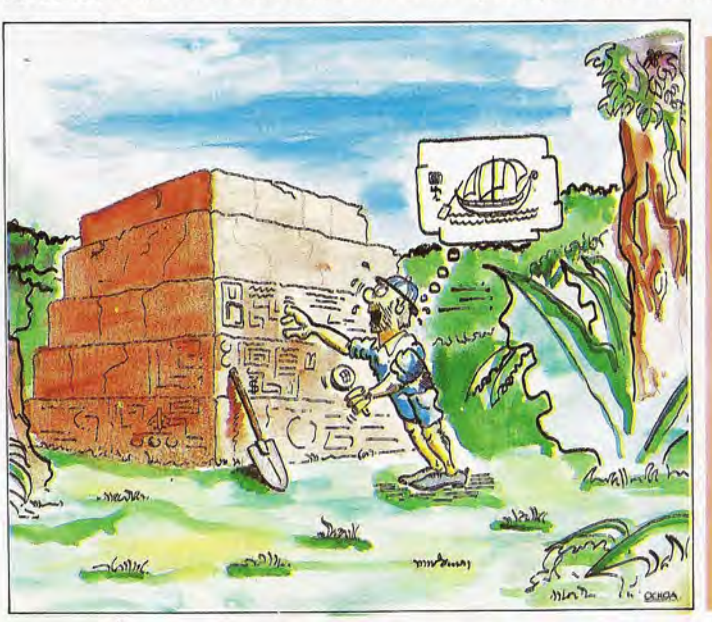
Un paso posterior incorpora las llamadas *macroinstrucciones,* en las que los códigos de operación ya no coinciden exactamente con los de máquina y la descomposición del problema no tiene por qué llegar al nivel más elemental.

Se dice que los lenguajes de ensamble son próximos a la máquina porque siguen la estructura de sus instrucciones y cada tipo de ordenador tiene su propio lenguaje de ensamble. No resuelven el problema de la incompatibilidad entre las distintas máquinas.



Lenguajes próximos al problema

Los lenguajes de este nivel resuelven principalmente el inconveniente señala-



Los programas escritos en lenguaje de máquina sólo pueden ser ejecutados por el ordenador que entiende ese lenguaje. Al igual que una inscripción jeroglífica solo es descifrable por un egiptólogo, un programa escrito en el código de una determinada máquina es ininteligible para las demás.

do en tercer lugar, ya que al alejarse de la máquina y aproximarse al problema no se encuentran ligados a ningún ordenador. Estos lenguajes, llamados de alto nivel, pueden ser utilizados en diferentes tipos de ordenadores.

Evidentemente, las instrucciones de los lenguajes de alto nivel son muy distintas de las elementales de la máquina, por lo que, en general, una instrucción de alto nivel realiza el mismo proceso que muchas instrucciones elementales de nivel máquina. El inconveniente apuntado en segundo lugar también es resuelto por los lenguajes de alto nivel, aunque siempre es necesario un mínimo conocimiento de las posibilidades del ordenador que estamos utilizando.

La traducción

Si la unidad de control sólo procesa instrucciones escritas en su propio lenguaje, a base de unos y ceros..., ¿cómo es posible que pueda trabajar con un programa escrito en un lenguaje tan alejado de su estructura?

Ello es posible gracias al propio concepto de ordenador, ya que un proceso de ordenador implica que un programa almacenado, utilizando unos datos de entrada, dé lugar a una información resultante en la salida.

¿Qué pasaría si los datos de entrada fueran nuestro programa, escrito en



Con los lenguajes de alto nivel la programación de los ordenadores no exige un profundo conocimiento de su estructura interna, con lo que cualquier usuario no especializado en la arquitectura íntima de los ordenadores puede llegar a confeccionar programas plenamente operativos.

cualquier lenguaje, y el programa almacenado fuera una secuencia completa de instrucciones para su traducción? El resultado sería un programa escrito en lenguaje de máquina.

La solución es análoga a la que se aplica en una conferencia internacional. Si un conferenciante habla en una lengua que no entiende el auditorio, se resuelve el problema mediante un *traductor.*

En informática se denomina *programa fuente* a un programa escrito en un lenguaje de ensamble o de alto nivel, y *programa objeto* al escrito en código máquina. Por tanto, un *programa objeto* sólo puede ser ejecutado en el ordenador correspondiente.

Un programa fuente podría ser ejecutado en cualquier ordenador si previamente se procede a su traducción, recurriendo al programa ensamblador o compilador correspondiente a la máquina en cuestión.

Para procesar datos con un programa de alto nivel es necesario realizar dos pasos:

1. Una vez almacenado el *programa traductor,* hay que cargar como datos el *programa fuente,* para obtener como resultado el *programa objeto* en código máquina.

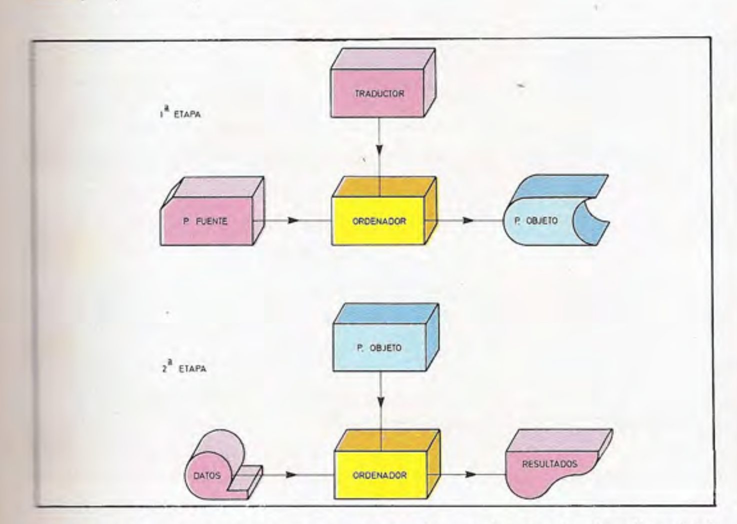
2. Ejecutar el *programa objeto* (resultante del proceso de traducción anterior), con lo que al alimentar los *datos* del problema se obtendrán los *resultados* buscados.



Los lenguajes de alto nivel se parecen más al idioma que hablan los hombres de negocios, técnicos y científicos que al de la propia máquina.



Los ensambladores y compiladores convierten los «programas fuente» en «programas objeto» descifrables por el ordenador. El proceso de traducción corre a cargo del propio ordenador, auxiliado por los oportunos programas traductores (ensamblador o compilador).



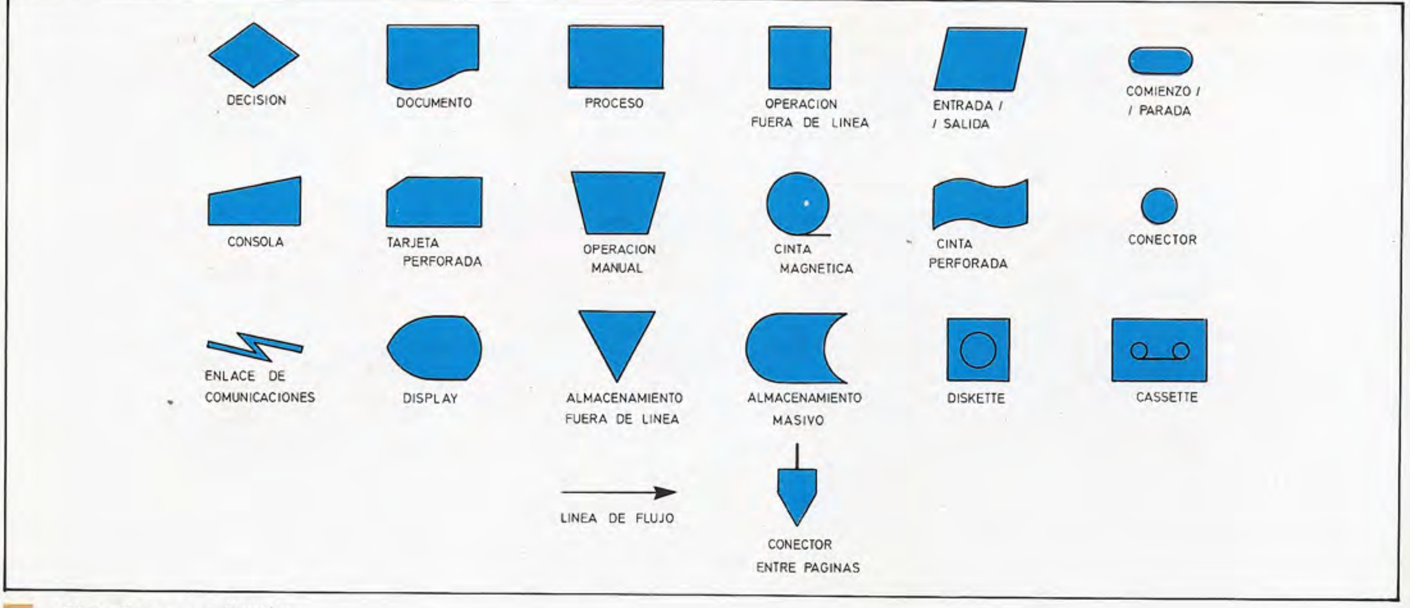
Para ejecutar un programa escrito en lenguaje de alto nivel se requieren dos etapas. En el ejemplo, un «programa fuente» perforado en tarjetas, se convierte en «programa objeto» almacenado en disco. En la segunda etapa, el programa objeto es ejecutado por el ordenador.



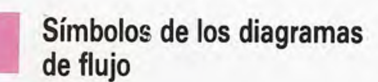
El ordenador puede encargarse de traducir los programas escritos en lenguaje de alto nivel a programas en código máquina. Ello lo consigue ejecutando un programa «traductor» que utilice como datos a procesar las instrucciones del programa fuente.

Compatibilidad de programas

En teoría, todo programa escrito en un lenguaje de alto nivel podría ejecutarse en cualquier ordenador si se dispone del traductor adecuado. En la práctica no siempre es así, ya que tanto los fabricantes de máquinas como los diseñadores de compiladores introducen limitaciones y modificaciones. Debido a ello, para pasar un programa de un ordenador a otro es necesario realizar algunos cambios en el formato de determinadas instrucciones.



Símbolos normalizados de los diagramas de flujo.



Símbolos de los diagramas de flujo

Los símbolos de los diagramas de flujo surgen para mostrar, de una manera gráfica y fácilmente reconocible, los pasos que se siguen en un proceso de ordenador. En realidad, cada usuario de ordenador podría tener sus propios símbolos para representar sus procesos en forma de diagrama de flujo. Esto supondría que sólo él, que conoce sus símbolos, estaría en condiciones de interpretarlos. Para resolver este problema y hacer comprensibles los diagramas a todas las personas, los símbolos se sometieron a una normalización. No vamos a mostrar aquí todos los símbolos de los diagramas de flujo, pero sí hablaremos de los más utilizados.

1. **Funciones de proceso**

Proceso: Cualquier función de proceso realizada por el ordenador. Por ejemplo, sumar dos cantidades.
Operación manual: Cualquier operación manual realizada «fuera de la línea», pero no por un equipo automático. Ejemplo de un símbolo de este tipo sería el de perforación de tarjetas.
Operación por equipo fuera de línea: Cualquier operación «fuera de línea» que no dependa de la velocidad humana, tal como el efectuado por una unidad de microfilm.

2. **Funciones de entrada/salida y archivo**

Tarjeta perforada: Los datos de E/S están en tarjetas perforadas.
Cinta magnética: Los datos de E/S se encuentran en cinta magnética.
Casete: Los datos de E/S se encuentran grabados en una cinta de casete.
Disquete: Los datos de E/S están en un disco flexible.
Cinta de papel perforado: Los datos de E/S se encuentran en cinta de papel perforado.
Documento: Normalmente es una salida; aunque puede representar una entrada en los casos de caracteres ópticos (OCR) y magnéticos (MICR).
Almacenamiento masivo: Generalmente disco magnético, aunque también puede indicar tambor magnético u otro medio de archivo.
Visualización (DISPLAY): En general, una pantalla CRT (tubo de rayos catódicos).
Entrada manual: Normalmente un teclado que permite la entrada de datos. También se usa como salida cuando el terminal es de teletipo.

3. **Conexiones**

Línea de flujo: Une dos símbolos.
Enlace de comunicaciones: Este símbolo indica el medio de transmisión entre elementos remotos de un equipo informático.
Conector entre páginas: Lo mismo que el anterior, pero los dos puntos de unión se encuentran en páginas diferentes.

4. **Otros símbolos**

Decisión: Para determinar cuál de los varios caminos posibles puede seguirse.
Comienzo o fin: Indica el comienzo o el final de un proceso.

El lenguaje máquina

Dialogar con ceros y unos



La unidad central de proceso de todo operador posee un repertorio de instrucciones elementales que es capaz de reconocer y ejecutar. Este «código de máquina» es el que en última instancia permitirá programar los circuitos del ordenador para que realicen las tareas de proceso que el usuario decida encomendarle.

Cada instrucción de nivel de máquina contiene varios campos o elementos de información. Uno de ellos es el denominado código de operación, el cual indica a la unidad de control qué operación debe efectuar. El resto de la instrucción coincide con el operando, zona ésta que aporta el dato o datos a operar o, en ocasiones, una referencia a la posición de memoria en la que están ubicados los datos.

Ciertos formatos de instrucción incluyen una zona suplementaria —que denominaremos indicador— con información relativa, por ejemplo, al dispositivo de E/S afectado por la instrucción.

Formatos de las instrucciones

Las instrucciones en código máquina de los diversos procesadores tienen un formato muy variado. Uno de los formatos más complejos es el que sigue:

CO	D10	D20	DR	DSI	I

en donde CO es el código de operación; D10 la dirección del primer operando; D20 la dirección del segundo operando; DR la dirección en la que se debe almacenar el resultado; DSI la dirección en la que se encuentra la siguiente operación, y, por último, I un indicador que especifica —cuando existe— algo más sobre la instrucción (periférico afectado, registro especial, etc.).

Este sería el caso de una instrucción de cuatro direcciones, aunque las hay también de tres; en todo caso, lo normal es que sean de dos o de una dirección, dependiendo de la arquitectura de la CPU.

En los microordenadores, las instrucciones de nivel máquina suelen constar de uno, dos o tres bytes. El primer byte contiene el código de operación, mientras que los otros dos contienen el dato o dirección del dato.

¿Cómo opera un procesador?

El procesador (microprocesador en el caso de los sistemas microordenadores) dispone de registros de apoyo para la ejecución del programa. Dos de los más relevantes son el contador de programa (en donde se almacena la dirección de la próxima instrucción que se debe ejecutar) y el registro de instrucción (en donde se decodifica la instrucción).

El contador de programa indica en qué dirección de la memoria se encuentra almacenada la instrucción. Una vez leída, el primer byte que contiene el código de operación es trasladado, a través del bus de datos, al registro de instrucción, donde es interpretado por el decodificador; acto seguido, el contador de programa se incrementa en una unidad. Si la instrucción contiene más bytes, el segundo pasa al registro de instrucción y se repite el proceso; y así sucesivamente.

Una vez que se han decodificado o interpretado todos los bytes, se ejecuta la instrucción. El proceso se repite para la instrucción siguiente: se traslada el nuevo código de operación al registro de instrucción, desde la posición de memoria apuntada por el contador, repitiéndo-

Instrucción de un byte	B7 B6 B5 B4 B3 B2 B1 B0	
Instrucción de dos bytes	B7 B6 B5 B4 B3 B2 B1 B0	D7 D6 D5 D4 D3 D2 D1 D0
Instrucción de tres bytes	B7 B6 B5 B4 B3 B2 B1 B0	D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
	Código operación	Dato o dirección

En general, existen tres formatos de instrucciones a nivel máquina: de uno, dos o tres bytes. El primer byte corresponde al código de operación y los restantes contienen el operando (dato o dirección del dato).



Al programar en lenguaje máquina «numérico» en su nivel más elemental, el programador debe escribir las instrucciones en lenguaje binario: ceros y unos.



Con un leve perfeccionamiento, consistente en la inclusión de un codificador, la tarea de programación en lenguaje máquina puede facilitarse al permitir la escritura de las instrucciones en hexadecimal.

se el proceso hasta que se llega al final del programa.

¿Programar en binario?

Dado que todo el proceso se realiza con bits, es natural que, en principio, el programa deba estar en lenguaje binario. El programador debe escribir en binario tanto el código de operación como los datos y direcciones, tal como resultaba inexcusable en los primeros ordenadores.

El siguiente paso consiste en utilizar el sistema hexadecimal o decimal para escribir las instrucciones. Así se simplifica la labor del programador.

Los códigos de operación y las direcciones en decimal se utilizaron en algunos ordenadores de la segunda generación, aunque hoy día lo usual es utilizar la representación hexadecimal. Estos lenguajes de máquina reciben el calificativo de *numéricos* para distinguirlos de los lenguajes de máquina *simbólicos*. Estos últimos sustituyen el código de operación numérico por un código alfabético que es nemotécnico, es decir, que le recuerda al operador lo que hace la instrucción. Es más fácil recordar que ADD significa sumar (sobre todo si se tienen conocimientos elementales de inglés) que retener en la memoria que para sumar hay que emplear el código hexadecimal 09 (adición en el microprocesador Z80).

Siempre que se programe en el lenguaje propio de la máquina, el programador debe llevar el control de las posiciones de memoria en las que almacena los diferentes datos, es decir, necesita utilizar el llamado mapa de direcciones de memoria.

Tipos de instrucciones

Las instrucciones se suelen agrupar en diversas categorías que determinan la naturaleza de la tarea que se ordena a la CPU. Los principales tipos son:

- *Instrucciones de transferencia de datos*

Permiten la lectura o escritura desde o hacia la memoria y entre registros internos del procesador. Incluyen también la carga y descarga de registros (de memoria a acumulador o viceversa, etc.).

- *Instrucciones de ruptura de secuencia*

Son las instrucciones que realizan los saltos y las bifurcaciones de una parte a otra del programa. Pertenecen a este grupo tanto los saltos incondicionales como los condicionales, las instrucciones de control de bucles, las llamadas a subrutinas, las de retorno al programa principal, etc.

- *Instrucciones de entrada/salida*

Relacionan a los periféricos con la unidad central de proceso y la memoria.

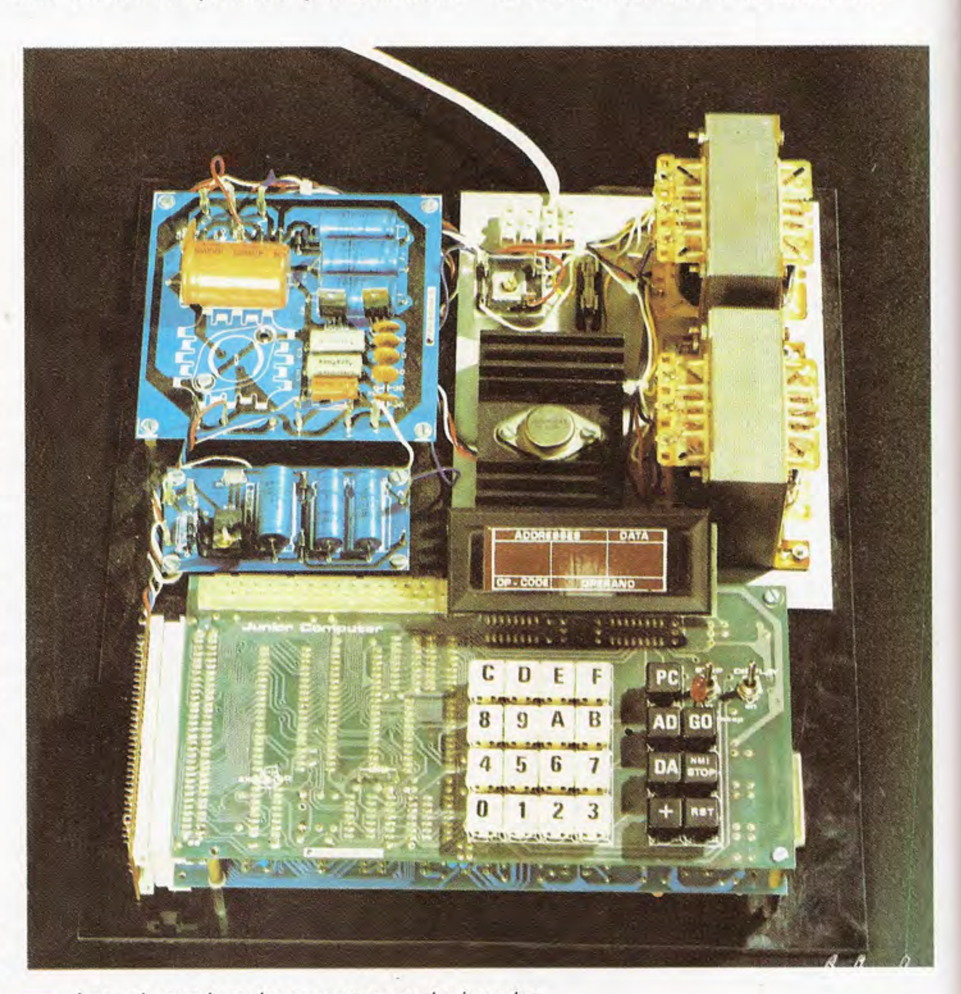
- *Instrucciones de control*

Permiten controlar el programa y el propio sistema. Comprenden el control del *status* (registro de estado), la no operación, la parada, etc.

Los conjuntos de instrucciones máquina, así como sus códigos nemotécnicos, difieren de un equipo a otro, por lo que es necesario que el programador disponga de la tabla correspondiente.

Direccionamientos

Los bytes que siguen al código de operación corresponden al operando: un dato, una dirección o un indicador.



Los microordenadores menos evolucionados suelen programarse en el lenguaje máquina «numérico» propio del microprocesador que constituye su CPU. La representación numérica de la información suele realizarse en el sistema hexadecimal.

El código de operación señala a la unidad de control el tipo de direccionamiento utilizado y, por consiguiente, dónde localizar el dato correspondiente. Los tipos de direccionamiento más usuales son:

- *Direccionamiento implícito*

Opera entre registros internos. Son instrucciones de un solo byte. Ejemplo: RTS (60) del microprocesador 6502, que corresponde a retorno de subrutina.

- *Direccionamiento inmediato*

Se opera directamente con el dato contenido en los bytes de operando. Ejemplo: ADI (C5) del 8080; suma al acumulador el segundo byte de la instrucción.

- *Direccionamiento directo*

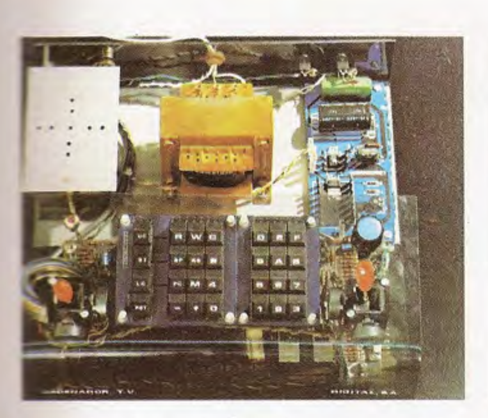
Hay que acceder a la posición indicada por los bytes de operando; en esa posición de memoria se encuentra el dato a operar. Ejemplo: SUB del 8085; resta del acumulador el contenido del registro cuya dirección está en los bytes de operando.

- *Direccionamiento relativo*

El dato hay que tomarlo de la posición de memoria cuya dirección es la apuntada por el contador del programa más o menos un número indicado por el segun-



En los lenguajes de máquina simbólicos, el programador utiliza códigos nemotécnicos en lugar de códigos numéricos. En cualquier caso, sigue siendo necesario llevar un mapa de direcciones reales en las que se van almacenando los datos.



Una de las dificultades inherentes a la programación en lenguaje máquina es que los programas sólo son ejecutables en equipos análogos, basados en la misma unidad central de proceso.



Los innumerables alfabetos que existen sirven para representar de forma simbólica los lenguajes que permiten la comunicación.

Introducción a la teoría de los lenguajes

Noam Chomsky inició el estudio de los lenguajes formales al crear un modelo matemático de una gramática en 1956. Hoy día el estudio de las gramáticas formales es uno de los campos más importantes de la informática teórica.
Webster define el lenguaje como «el conjunto de palabras y reglas para combinarlas, que es usado y entendido por una comunidad numerosa». Pero esta definición clásica no es rigurosa matemáticamente. Cuando los seres humanos usan un lenguaje, se puede permitir la existencia de palabras, términos o frases con significado simbólico o ambiguo, ya que la inteligencia y el buen juicio del oyente le permite comprender el sentido exacto que quiere darle el emisor, soslayando las discrepancias que puedan existir entre la forma y el fondo, esto es, entre la sintaxis y la semántica.
Para que una máquina entienda el significado concreto de un mensaje necesita que se le comunique en un lenguaje que esté rigurosamente definido con unas estrictas reglas gramaticales.
La comunicación hombre-máquina se debe realizar a través de un lenguaje escrito que impida la posibilidad de error en la interpretación de los mensajes.
Los conceptos fundamentales de la teoría de lenguajes son los de alfabeto, cadena, lenguaje y gramática.
Se llama *alfabeto* a un conjunto no vacío de símbolos gráficos.
Ejemplos típicos son:
El alfabeto castellano = { ⓐ ⓑ ⓒ ... ⓩ }.
El alfabeto griego = { ⓐ ⓑ ⓒ ... ⓩ }.
El alfabeto binario = { ⓪ ① }.
Hemos encerrado los elementos del alfabeto en círculos para distinguirlos, ya que ni la coma ni el espacio en blanco indicarían la separación, puesto que éstos pueden ser elementos del lenguaje (el espacio o la coma tiene ese carácter en muchos lenguajes de ordenador). No obstante, lo normal es representar los elementos espaciados o separados por comas.
A los alfabetos se les suele llamar también *conjuntos de base* o *vocabularios* y se les representa por los símbolos Σ o V. Hay alfabetos que no son gráficos, como el alfabeto musical, formado por las notas (aunque luego se las represente gráficamente mediante símbolos en un pentagrama), el de los sordomudos o el de los ciegos.

do byte del operando. Ejemplo: BBC (90) del 6502; salta a la dirección del contador más el valor del segundo byte, atendiendo a un determinado indicador del registro de estado.

- *Direccionamiento indexado*

La dirección del dato se obtiene sumando el valor del registro indicado al valor de la dirección absoluta incluida en la instrucción. Ejemplo: DEC (D6) del 6502; resta una unidad al valor que está en la posición de memoria apuntada por el contenido del registro índice X más el valor del segundo byte de la instrucción.

Al igual que ocurre con los repertorios de instrucciones máquina, los tipos de direccionamiento de las instrucciones difieren de un equipo a otro y dependen de la CPU utilizada.

¿Hay ventajas al programar en lenguaje de máquina?

El conocimiento exhaustivo de las instrucciones (códigos y modos de direccionamiento), así como de las direcciones reales de memoria en donde se almacenan los datos, es una de las principales dificultades para programar en lenguaje de máquina. Además, los programas en código de máquina no son ejecutables en otro equipo que no tenga un procesador igual o compatible.

Entonces, ¿es útil conocer el lenguaje máquina del equipo, aun en el caso de que éste admita la programación en un lenguaje de alto nivel? El profesional de la programación sí debe conocerlo por las siguientes razones:

— Si se tiene poca capacidad de memoria, siempre se podrá recurrir al lenguaje de máquina que es el que menos memoria ocupa.

— Una rutina que se haya de utilizar en múltiples ocasiones, programada en código de máquina no sólo ahorra memoria, sino también tiempo de ejecución.

— Las mejoras o modificaciones en el software de base son más efectivas si se hacen en lenguaje de máquina; hay que recordar que los programas en código de máquina son los que se ejecutan en menos tiempo.

```
0010:                    REPEAT ROUTINE
0020:
0030: 0000                      ORG   $0000
0040:
0050:                    TEMPORARY DATABUFFERS IN PAGE ZERO
0060:
0070: 0000               KEY    *     $00DA
0080: 0000               NOTEL  *     $00DC
0090: 0000               NOTEH  *     $00DD
0100: 0000               LENGTH *     $00DE
0110:
0120:                    INTERVAL TIMER
0130:
0140: 0000               CNTA   *     $1AF4  DISABLE TIMER IRQ
0150: 0000               CNTD   *     $1AF7  DISABLE TIMER IRQ, CLK1KT
0160: 0000               CNTG   *     $1AFE  ENABLE TIMER IRQ, CLK64T
0170: 0000               RDFLAG *     $1AD5  B7 IS TIMER FLAG
0180:
0190:                    GOTO MONITOR
0200:
0210: 0000               RESET  *     $1C1D  NEW I/O DEFINITION
0220:
0230:                    I/O DEFINITION
0240:
0250: 0000               PBD    *     $1A82
0260: 0000               PBDD   *     $1A83
0270:
0280:                    IRQ VECTOR
0290:
0300: 0000               IRQL   *     $1A7E
0310: 0000               IRQH   *     $1A7F
0320:
0330:
0340:                    START OF THE REPEAT PROGRAM
0350:
0360: 0000 78            REPEAT SEI          DISABLE IRQ LINE
0370: 0001 D8                   CLD
0380: 0002 A9 30                LDAIM IRQRE  SET UP IRQ VECTOR
0390: 0004 8D 7E 1A             STA   IRQL
0400: 0007 A9 1A                LDAIM IRQRE  /256
0410: 0009 8D 7F 1A             STA   IRQH
0420: 000C A9 01                LDAIM $01    PB0 IS OUTPUT
0430: 000E 8D 83 1A             STA   PBDD
0440: 0011 8D 82 1A             STA   PBD    TOGGLE SPEAKER OFF
0450: 0014 85 DD                STAZ  NOTEH  SET NOTE POINTER
0460: 0016 A9 00                LDAIM $00
0470: 0018 85 DC                STAZ  NOTEL  SET NOTE POINTER
0480: 001A 8D F4 1A             STA   CNTA   RESET IRQ LINE, DISABLE TIMER IRQ
0490: 001D 58                   CLI          ENABLE CPU IRQ
0500:
0510: 001E A9 FF         FETCH  LDAIM $FF    SET TIMER ENABLE TIMER IRQ
0520: 0020 8D FE 1A             STA   CNTG
0530: 0023 A0 00                LDYIM $00    FETCH NOTE
0540: 0025 B1 DC                LDAIY NOTEL
0550: 0027 85 DA                STAZ  KEY
0560: 0029 C8                   INY          FETCH LENGTH
0570: 002A B1 DC                LDAIY NOTEL
0580: 002C 85 DE                STAZ  LENGTH
0590: 002E A4 DA                LDYZ  KEY    LOOKUP CONVERSION
0600:
0610: 0030 A9 00         TONE   LDAIM $00    TOGGLE SPEAKER ON
0620: 0032 8D 82 1A             STA   PBD
0630: 0035 BE 00 1A             LDXY  DEL    GET FREQUENCY
0640: 0038 20 70 00      TONEA  JSR   EQUALA DELAY 22 MICRO SEC
0650: 003B CA                   DEX
0660: 003C D0 FA                BNE   TONEA  LOOP TIME IS 27 MIKRO SEC*X
0670: 003E A9 01                LDAIM $01    TOGGLE SPEAKER OFF
0680: 0040 8D 82 1A             STA   PBD
0690: 0043 BE 00 1A             LDXY  DEL    GET FREQUENCY AGAIN
0700: 0046 A5 DE         TONEB  LDAZ  LENGTH GET LENGTH
0710: 0048 30 08                BMI   TONEC  TIME OUT?
0720: 004A 20 74 00             JSR   EQUALB EQUALIZE 17 MICRO SEC
0730: 004D CA                   DEX
0740: 004E D0 F6                BNE   TONEB  LOOP TIME IS 27 MICRO SEC*X AGAIN
0750: 0050 F0 DE                BEQ   TONE   RETURN AFTER ONE PERIODE
0760: 0052 A2 04         TONEC  LDXIM $04    LOOP TIME = 4*CNTD*PRESET
0770: 0054 A9 30         TONED  LDAIM $30    PRESET = $30
0780: 0056 8D F7 1A             STA   CNTD   DISABLE TIMER IRQ
0790:
0800: 0059 2C D5 1A      POLL   BIT   RDFLAG READ FLAG REGISTER, TIME OUT?
0810: 005C 10 FB                BPL   POLL   IS TIMER FLAG STILL ZERO?
0820: 005E CA                   DEX
0830: 005F D0 F3                BNE   TONED  LOOP COUNTER ZERO?
0840: 0061 E6 DC                INCZ  NOTEL  ADJUST NOTE POINTER
0850: 0063 E6 DC                INCZ  NOTEL
0860: 0065 A0 00                LDYIM $00
0870: 0067 B1 DC                LDAIY NOTEL  END OF NOTE BUFFER?
0880: 0069 C9 77                CMPIM $77    EOF CHARACTER
0890: 006B D0 B1                BNE   FETCH  IF NOT EOF, CONTINUE
0900: 006D 4C 1D 1C             JMP   RESET  ELSE BACK TO MONITOR
0910:
```

Listado de un programa confeccionado en lenguaje máquina para el microprocesador 6502.

Lenguajes «de ensamble»

Entre la intimidad de la máquina y el problema



Para evitar la necesidad de utilizar códigos numéricos y direcciones reales de memoria al programar, se desarrollaron los lenguajes de ensamble, también conocidos como lenguajes simbólicos, lenguajes ensambladores o «assemblers».

Estos lenguajes permiten escribir los programas representando los diferentes elementos de forma simbólica. Los símbolos utilizados son de dos tipos: fijos para los códigos de operación y variables para las direcciones.

Como cada procesador sólo tiene un juego de instrucciones máquina ejecutables por su unidad de control, los lenguajes ensambladores son específicos para cada ordenador.

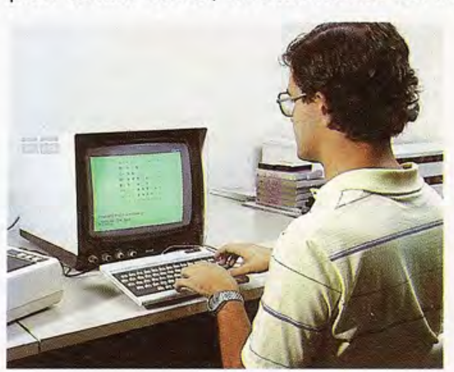
El campo de código de operación contiene el nombre nemotécnico asignado a la instrucción. Estos códigos son elementos fijos del lenguaje y el programador no puede ni alterarlos ni usarlos como etiquetas u operandos. En el campo de operandos se codifican los identificadores de dirección de los mismos. Hay instrucciones que no tienen operando, aunque, por lo general, éste se compone de uno o más valores separados por comas. También pueden darse valores constantes o expresiones aritméticas.

El campo de comentario no es obligatorio y es ignorado durante la fase de traducción. Sirve exclusivamente para facilitar la lectura del programa fuente a cualquier persona.

Las instrucciones corresponden habitualmente a los tipos ya estudiados en otros apartados, si bien los lenguajes de ensamble añaden un nuevo tipo: las pseudoinstrucciones.

Las pseudoinstrucciones

Su nombre procede de que se utilizan para definir datos, codificándose de la



Los lenguajes ensambladores ocupan el nivel inmediatamente superior al de los lenguajes máquina; permiten escribir los programas representando los diversos elementos de forma simbólica.

Lenguajes simbólicos simples

Los lenguajes tienen una sintaxis cuyas leyes dependen de la estructura de la máquina y de las restricciones impuestas a la redacción de las instrucciones.

Los programas fuente aparecen como un conjunto de líneas, denominadas *sentencias,* que generalmente contienen un código de operación simbólico.

La estructura de una sentencia suele ser:

Etiqueta-Código-Operandos-Comentario

Algunos ensambladores tienen el formato fijo, distribuyéndose las zonas en posiciones prefijadas de la línea de escritura. Otros son de formato libre, bastando con respetar el orden de los elementos y establecer una separación entre ellos.

La *etiqueta* es el identificador de la dirección de la instrucción. Puede ser elegida libremente por el programador. En general es alfabética, aunque a veces puede incluir cifras siempre que el primer carácter sea una letra. No es necesario etiquetar todas las instrucciones; sólo se etiquetarán aquellas a las que haya que referenciar en otro punto del programa.

```
02160          RNTY   XI-9
02170          TFLS   ZZ.DT
                                      00702 36 01184 00100
                                      00714 16 01425 00737
                                      00726 49 01372 00000
                                      00733 00005    01227
                                      00738 00005    01238
02180 RST1     BNR    OPER.DT + 1,7
                                      00744 45 00792 01239
02190          TBTY
                                      00756 34 00000 00108
02200          WNTY   ZZ-9
                                      00768 38 01218 00100
03010          B      REST
                                      00780 49 00888 00000
030200 PER     BTFS·  POR.X1
                                      00792 16 01423 00815
                                      00804 49 01392 00000
                                      00811 00005    00960
                                      00816 00005    01193
03030          AM     RST1 + 11,10
                                      00822 11 00755 00010
03040 S''M     BTFS   MAS.DT + 10
                                      00834 16 01423 00857
                                      00846 49 01392 00000
                                      00853 00005    01014
                                      00858 00005    01248
03050          AM     S''M + 28,10
                                      00864 11 00862 00010
03060          B      RST1
                                      00876 49 00744 00000
```

Listado obtenido al final de un proceso de «ensamblado». En el mismo se observan las diversas instrucciones en código máquina generadas por cada macroinstrucción.

misma forma que las instrucciones de máquina, aunque no lo sean.

En general, son sentencias que no producen directamente ningún código objeto, sino que ayudan a la definición del resto del programa, complementando la codificación. Suelen conservar los mismos símbolos en los diferentes lenguajes, siendo los principales tipos los siguientes:

De principio y fin de programa: sirven para el control de la traducción.

De definición de constantes: se emplean para introducir constantes y referirse a ellas por medio de un identificador.

De reserva de zona de memoria: útiles, por ejemplo, para cargar tablas, matrices, etc.

Ventajas y desventajas de los lenguajes de ensamble simples

Entre las principales ventajas se encuentran la reducción de los errores lógicos (puesto que no se emplean direcciones reales), la fácil eliminación de los errores formales (ya que son detectados en la traducción) y la disminución de los tiempos de programación. Tienen el inconveniente de que cada ordenador posee un lenguaje ligado a su estructura y juego de instrucciones, por lo que el programador tendría que conocer diversos lenguajes ensambladores si quiere trabajar en distintas máquinas.

Lenguajes autocodificadores

La limitación de usar sólo el conjunto de instrucciones de que dispone una máquina es eliminada por un nuevo nivel de lenguajes, los llamados lenguajes de nivel *autocodificador, macroensamblador* o *macroprocesador,* que constituyen el primer paso hacia la independencia entre el lenguaje y la máquina gracias a la introducción de las *macroinstrucciones.*

Una *macroinstrucción* es una instrucción que no se corresponde directamente con una instrucción del lenguaje de máquina, sino que representa operaciones que pueden desglosarse en secuencias más o menos largas de instrucciones máquina.

Las macroinstrucciones se componen en general de dos campos: el *campo de operación* y el *código paramétrico,* cumpliendo el primero el mismo papel que el código de operación, mientras que el campo paramétrico contiene los datos con los que se realiza la macroinstrucción. Estos datos pueden ser numéricos o simbólicos. Las macroinstrucciones permiten programar, en una sola instrucción, operaciones tales como la comparación, división, etc.

Existen dos tipos de macroinstrucciones: *del lenguaje* y *del programador.* Las macroinstrucciones del lenguaje son aquellas que son proporcionadas por el constructor y, al igual que las microinstrucciones, tienen un código prefijado.

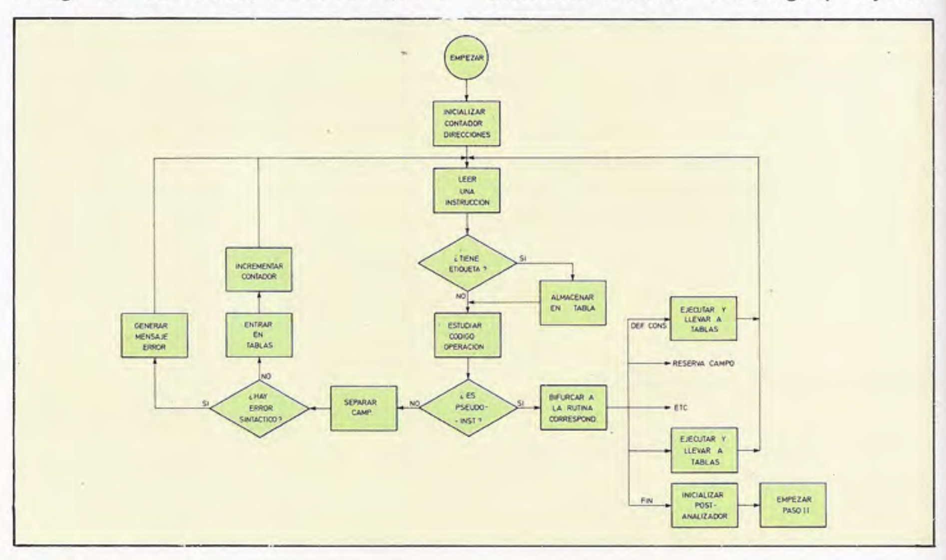


Diagrama de flujo detallado del primer paso de una operación de traducción.

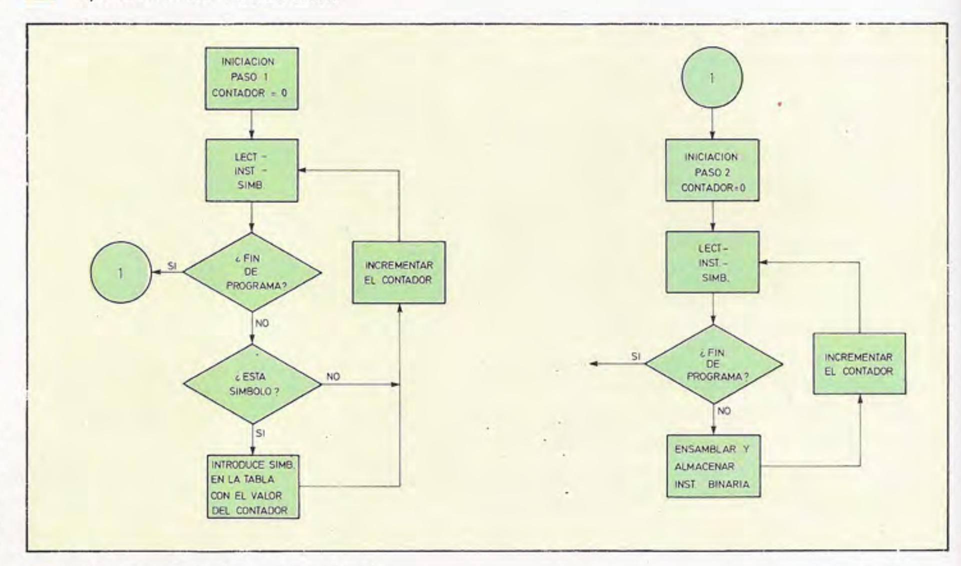


Diagrama de la secuencia de operaciones que realiza un programa ensamblador de dos pasos.

Macros del programador

Algunos lenguajes permiten que el programador cree sus propias «macros» mediante el uso de un *lenguaje de definición* de macros. Son muy útiles para introducir subprogramas.

Para que en la fase de traducción se pueda reconocer que se trata de una macroinstrucción de este tipo hacen falta dos pseudoinstrucciones: una de principio y otra de fin. Entre ambas se encuentra el cuerpo de la macroinstrucción, constituido por el conjunto de instrucciones que la desarrollan. En primer lugar se encuentra el *prototipo* de la macro, en el que aparecen el código elegido para la macro y los nombres simbólicos de los parámetros o argumentos.

Los lenguajes de nivel superior permiten la existencia de macroinstrucciones dentro de otras macroinstrucciones, e incluso la recursividad de la macroinstrucción, esto es, que una macroinstrucción se llame a sí misma, lo que puede ser útil, por ejemplo, para calcular el factorial de un número.

La principal ventaja de estos lenguajes es que no obligan a descender al nivel de instrucción elemental de máquina a la hora de programar. De todas formas, al hacer uso también de microinstrucciones, siguen siendo lenguajes próximos a la máquina, por lo que los programas escritos en estos lenguajes no se pueden procesar en todos los ordenadores.

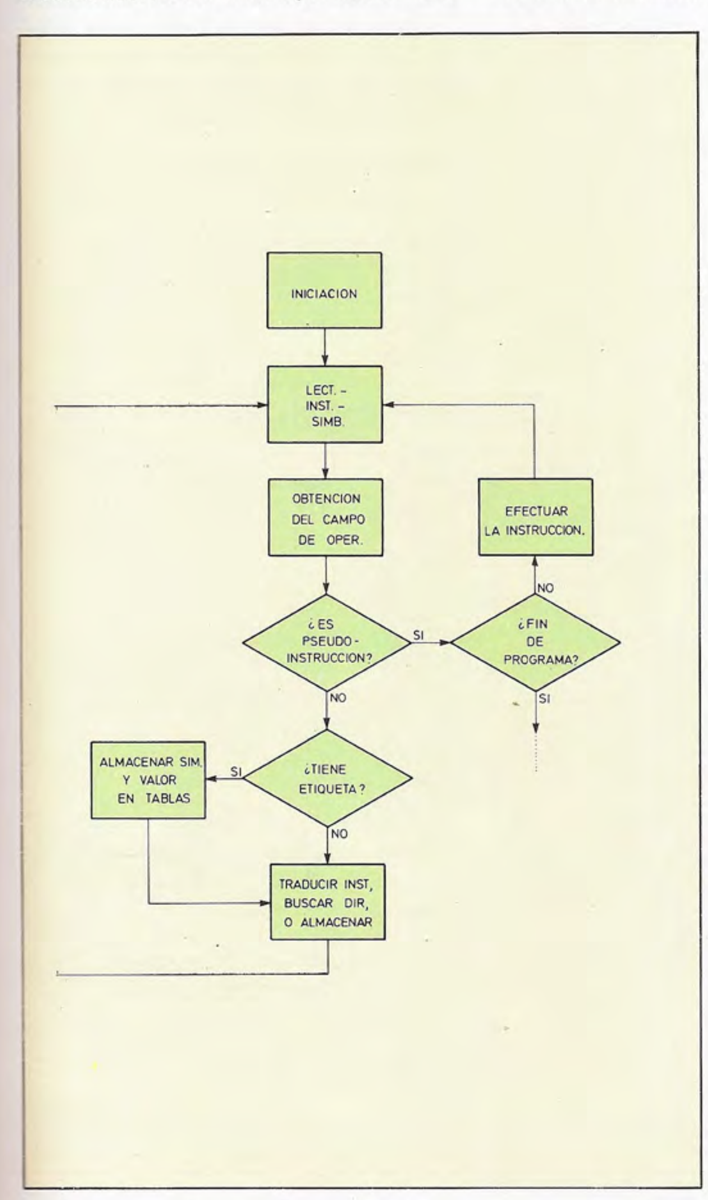
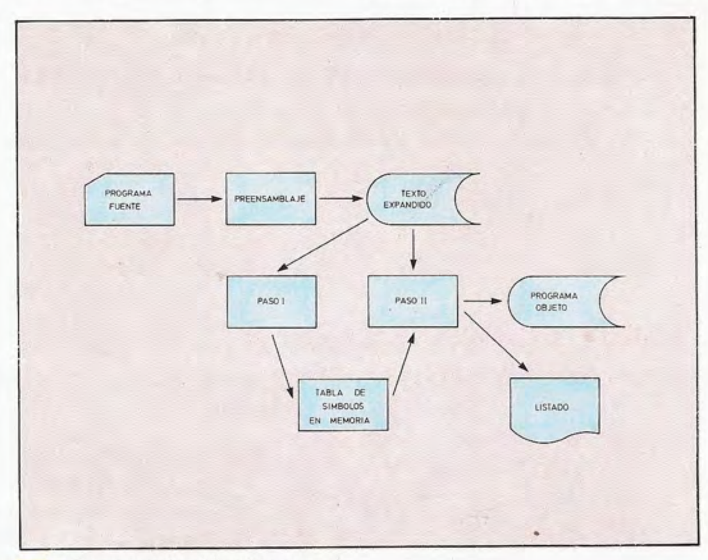


Diagrama de flujo asociado a los procesos de ensamblado «en un paso».



Representación de un proceso de macroensamblaje en tres pasos.



Aún a pesar de que son lenguajes más evolucionados que los de nivel máquina, los lenguajes de ensamble o ensambladores son específicos para cada máquina.

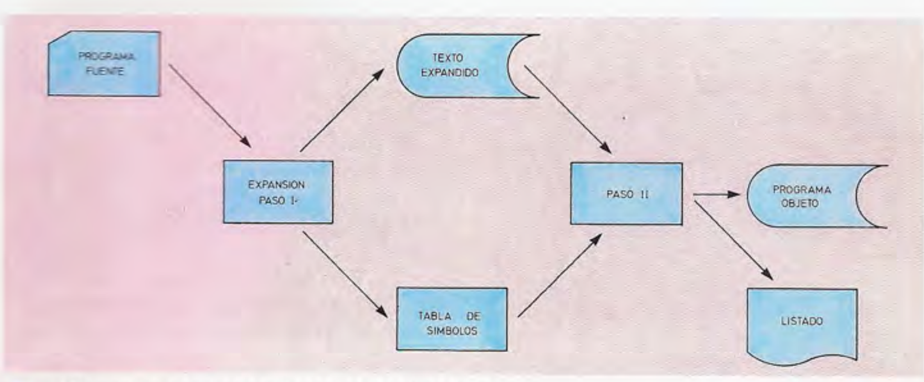


Diagrama correspondiente al desarrollo de un

Diagrama correspondiente al desarrollo de un proceso de macroensamblaje en dos pasos.

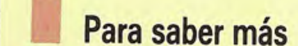
Etiqueta	Código	Operando	Comentarios
	LD	DT, 0	Poner la variable DT a 0
	LD	DI, 100	Poner la variable DI a 100
SUMT	ADD	DT, DI	Sumar DI a total DT
	SUB	DI, 1	Restar 1 a DI
	JR	NZ, SUMT	Continuar sumando si DI no es cero



El listado de la figura muestra un programa para la suma de los números decimales del cero al cien, confeccionado en lenguaje de ensamble o ensamblador.

Ensambladores

Los ensambladores son los encargados de convertir los programas fuente, escritos en lenguaje de ensamble, a programas objeto en código de máquina.
Como en todo proceso de traducción, junto con el programa objeto se generan normalmente los listados de errores sintácticos y de correspondencia entre el programa fuente y objeto.
El trabajo del ensamblador se reduce a una traducción palabra a palabra, cambiando por códigos de operación numéricos y direcciones reales los símbolos del programa. Para ello emplea tablas de traducción de símbolos, localizadas en la memoria, y lleva cuenta de la memoria ocupada.
Como a veces la definición de los símbolos puede venir detrás de la sentencia en la que aparecen por primera vez, la traducción se realiza repitiendo el proceso dos veces. Cada una se denomina *paso,* y de ahí que a este tipo de ensamblador se le conozca como *de dos pasos.*
En el primer paso construye la tabla de símbolos y realiza el análisis morfológico de las sentencias, detectando y escribiendo los errores sintácticos. En el segundo se genera el programa objeto, sustituyendo los símbolos por sus direcciones reales.
También existen *ensambladores de un paso* que realizan el proceso de una sola vez. Cuando un ensamblador de este tipo encuentra una sentencia que contiene un símbolo que todavía no está definido, la retiene en memoria, y según va encontrando definiciones vuelve hacia atrás y completa la traducción. De todas formas, su empleo práctico es bastante limitado.
La traducción de los programas escritos en lenguaje autocodificador o macroensamblador es realizada por los *macroensambladores;* éstos contienen además un ensamblador para traducir el resultado de la expansión de las macros.
Si se codifican todas las macros antes de cualquier llamada, se puede realizar la expansión junto con el primer paso de la traducción, obteniéndose un *macroensamblador de dos pasos.* En caso contrario hay que realizar la expansión en una fase anterior al ensamblaje (llamada *preensamblaje),* lo que da lugar a los *macroensambladores de tres pasos.*
El último avance en el campo de los ensambladores viene dado por los llamados *macroprocesadores de uso general* o *metaensambladores.*



Para saber más

¿Por qué se llama ensamblador?

Cuando un programador tiene que codificar un programa muy largo, lo divide en varios subprogramas o rutinas independientes que escribe, traduce y prueba por separado. Por consiguiente, el traductor debe seguir la pista de todas las referencias cruzadas, es decir, debe estar en condiciones de *ensamblar* todas las partes para dar un resultado único.

¿Es difícil aprender varios lenguajes de ensamble?

La mayoría de los lenguajes de esta clase son muy parecidos entre sí, por lo que cuando se conoce uno de ellos se pueden aprender otros sin una especial dificultad.

¿Qué se entiende por «expansión» de una «macro»?

Se llama expansión de una macroinstrucción al proceso que traduce una macro en la secuencia de instrucciones equivalente.

¿Qué son los parámetros o argumentos de una macro?

Se llaman argumentos o parámetros de una macro a los operandos de la misma. Reciben este nombre debido a la similitud entre macro y subprograma.

¿Qué es un metaensamblador?

Fue una idea de Ferguson basada en que los diferentes ensambladores tienen muchos puntos comunes. Un metaensamblador podría admitir la descripción de las reglas de ensamblaje para un ordenador particular, dando un resultado como si el trabajo hubiera sido realizado por el ensamblador específico. No están muy desarrollados en la actualidad.

Lenguajes de alto nivel

A un paso del lenguaje humano



Hoy en día, los lenguajes de alto nivel han alcanzado una profusión más que notable debido, principalmente, a que su estructura es muy próxima a la de los lenguajes naturales. Desde luego, el idioma del que deriva el vocabulario de esta categoría de lenguajes es el inglés, dado que la mayor parte de ellos han nacido en los Estados Unidos.

En teoría, los lenguajes de alto nivel no dependen del tipo de ordenador y pueden ser utilizados en diversas máquinas. En la práctica no siempre es así, sino que es necesario realizar ciertas modificaciones para llegar a disponer de un programa accesible en otro equipo distinto del de origen.

Características

Las características básicas de los lenguajes de alto nivel se presentaron en el capítulo anterior. A raíz de su evaluación, pueden deducirse toda una serie de ventajas e inconvenientes que determinan su interés real. Las ventajas más destacables son:

- Un programa escrito en un lenguaje de alto nivel puede ser utilizado —en algunos casos, después de someterlo a ligeras modificaciones— en distintos equipos.
- El tiempo de formación de los programadores es relativamente corto, en comparación con el necesario para aprender los lenguajes de nivel inferior.
- El programador no necesita conocer cómo funciona un ordenador específico para poder confeccionar los programas.
- El tiempo necesario para codificar y poner a punto un programa en lenguaje de alto nivel es inferior al necesario en el

El número de lenguajes informáticos se eleva día a día hasta el punto de que resulta casi imposible su catalogación. Ya en 1980 estaban registrados más de 200 lenguajes distintos, muchos de ellos con un elevado número de variantes o «dialectos».

caso de los lenguajes menos evolucionados.

- Los cambios y correcciones en los programas resultan más fáciles.
- Se reduce el coste de creación y mantenimiento de los programas.

A pesar de las virtudes relacionadas, también existen inconvenientes; los más significativos son:

- El incremento del tiempo de compilación.
- No se aprovechan plenamente las posibles ventajas de la arquitectura interna del sistema.
- Se incrementa la ocupación de memoria interna, tanto por parte del programa compilador como del programa objeto resultante.
- El tiempo de ejecución es mayor, puesto que las instrucciones generadas por el compilador son más numerosas que las correspondientes al mismo programa escrito directamente en código de máquina.

Clasificación

La clasificación de los lenguajes de programación próximos al problema es una misión casi imposible, debido a que cada día aparecen nuevos lenguajes y dialectos de los ya existentes. En 1980 estaban registrados unos 200 lenguajes diferentes, muchos de los cuales estaban especializados en la resolución de un determinado tipo de problemas o sólo eran adoptados en un reducido grupo de ordenadores.

Otra dificultad adicional está en determinar a qué categoría pertenece un lenguaje concreto. De una forma muy general podemos elaborar la clasificación que sigue, si bien las diversas categorías no son absolutamente disjuntas:

a) *Lenguajes científicos*

Históricamente son los primeros lenguajes evolucionados, debido a dos factores: en principio, la formulación matemática permite una más fácil formalización del lenguaje y, en segundo lugar, muchas de las apariciones científicas tienen un carácter poco repetitivo, por lo que resulta muy importante reducir el tiempo de programación.

Los primeros lenguajes fueron el SHORT CODE, creado por el doctor Mandy en 1949 para UNIVAC, y el SPEED CODING, desarrollado en 1953 por Backus y Seldon para IBM.

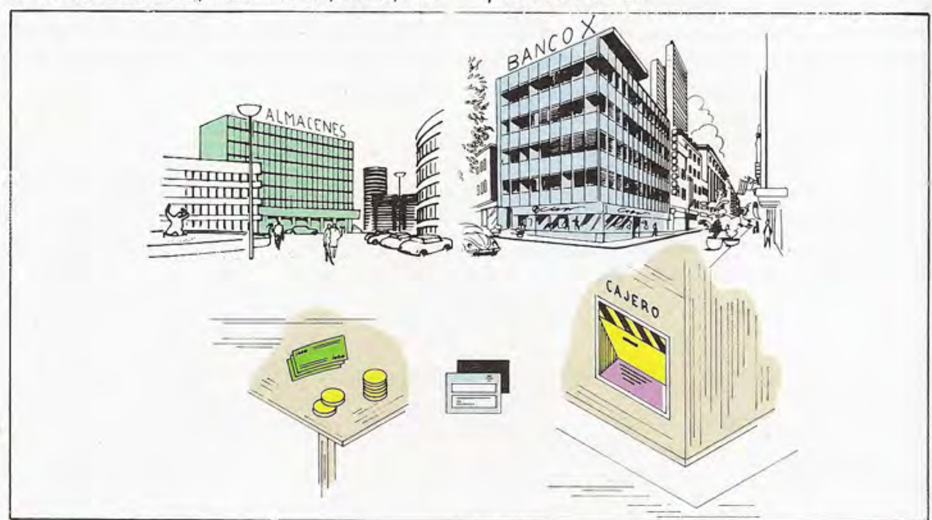
Antes de llegar el popular FORTRAN, aparecieron el MAHTMATIC, UNICODE, IT, GAT y FORTRANSIT.

Los lenguajes más conocidos, hoy en día, son: ALGOL, FORTRAN, APL, BASIC, PASCAL y C. De hecho, muchos de ellos se usan también en aplicaciones de gestión, como el BASIC o el PASCAL, aunque su origen es científico.

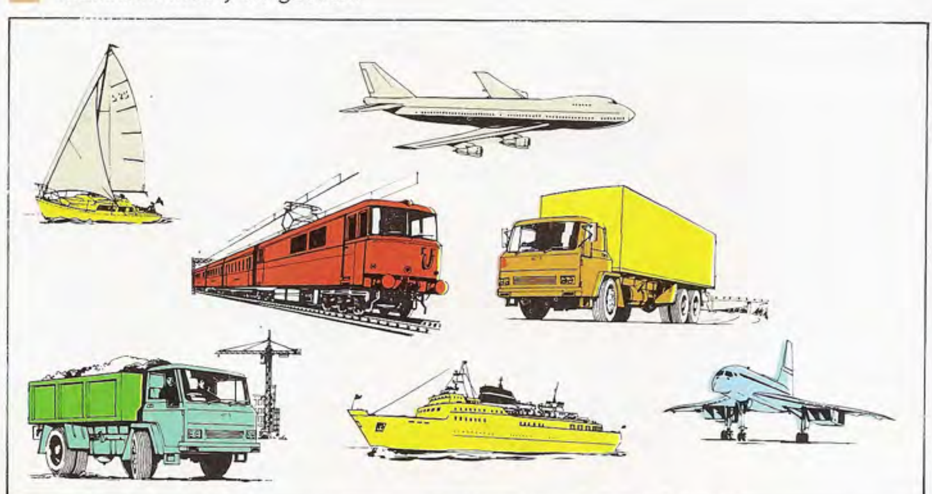
b) *Lenguajes de gestión*

Son lenguajes orientados a la solución de problemas de tratamiento de datos para gestión, por lo que predominan las instrucciones dedicadas a procesos de entrada y salida.

El primero fue el FLOW-MATIC, desarrollado en 1955 por el doctor Hopper para UNIVAC.



Los lenguajes orientados a aplicaciones de gestión son fundamentales en el mundo de los negocios. Esta categoría de lenguajes ha convertido a la informática en una herramienta imprescindible en los procesos administrativos y de gestión.



Los primeros lenguajes evolucionados que vieron la luz fueron creados para la programación de tareas científicas; su campo de aplicación se concreta en la creación de aplicaciones para investigación e ingeniería.

El lenguaje más característico entre los de gestión es el COBOL.

c) *Lenguajes polivalentes*

Son los resultados del intento de obtener un lenguaje que cubriera tanto el área científica como el área de gestión de una forma equilibrada.

El primero fue el JOVIAL, desarrollado en 1959 por el Strategic Air Command Control System.

Uno de los más conocidos es el PL/1, creado en 1964. Otros lenguajes de esta categoría son el FORMULA ALGOL, LISP2, LOGO, FORTH y ADA.

d) *Lenguajes para proceso de listas y cadenas*

Es un grupo muy especializado, entre los que cabe mencionar el IPL-V, el LISP1.5, el COMIT y el SNOBOL.

e) *Lenguajes para expresiones algebraicas formales*

Son lenguajes que permiten el uso de expresiones matemáticas sin referirse a valores numéricos concretos y, por tanto, no necesitan de un fuerte desarrollo de Análisis Numérico.

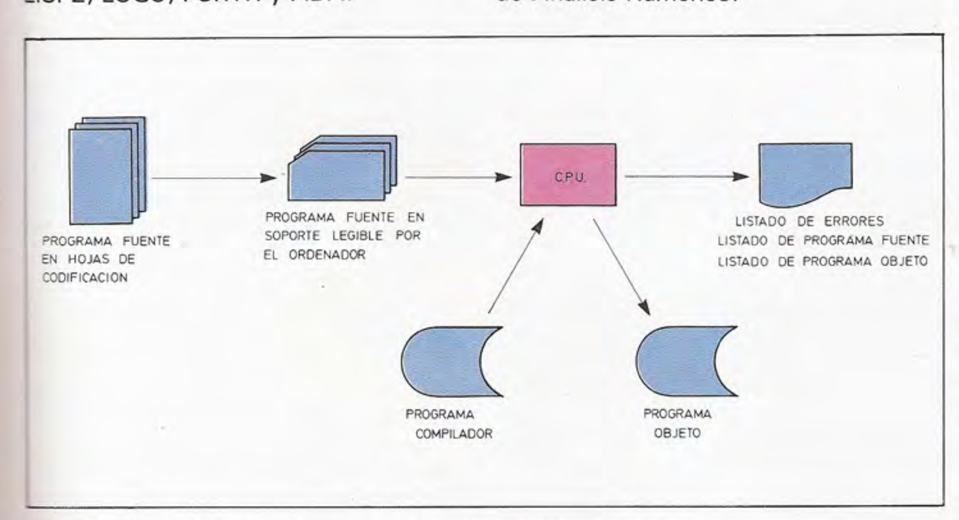
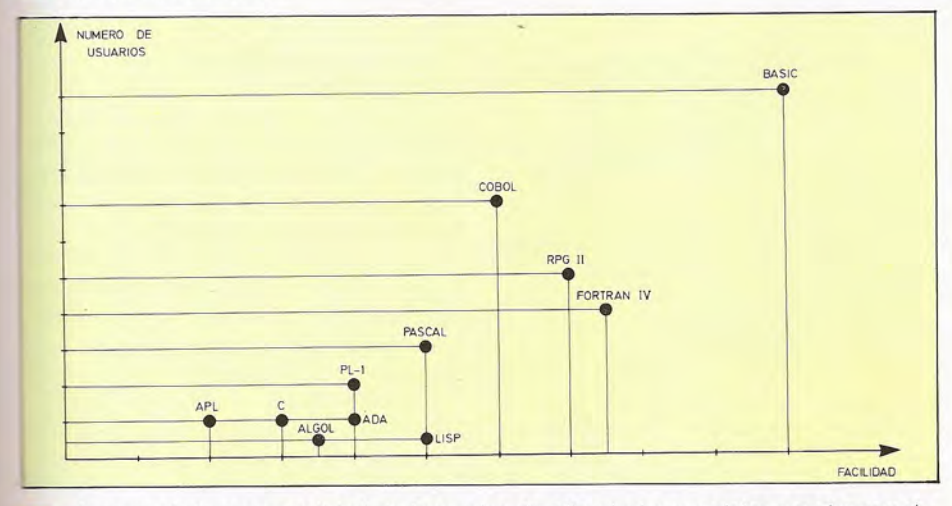


Diagrama de flujo del proceso de «compilación». Si se han producido errores, el sistema informará de tal circunstancia al usuario, que debe repetir el proceso hasta lograr una compilación correcta.



A pesar de la gran cantidad de lenguajes de alto nivel que existen en el mercado informático, el liderazgo corresponde a un grupo reducido. En el gráfico se relacionan las características de «facilidad de aprendizaje» y «número de usuarios» propias de los lenguajes más comunes.

Compiladores e intérpretes

Los *compiladores* son programas especializados en la traducción a código máquina de programas escritos en lenguaje de alto nivel.

Para ejecutar un programa escrito en un lenguaje de alto nivel es necesario, por tanto, compilarlo primero.

Con la profusión de los procesos interactivos ha nacido otro método de traducción; éste es el que ponen en práctica los denominados *intérpretes.*

Tal como evidencia su denominación, este tipo de programas efectúan la traducción y ejecución sucesiva, instrucción a instrucción (frase a frase). En consecuencia, se distinguen de los compiladores en que estos últimos traducen el programa completo, sin operar su ejecución a medida que avanza el proceso de traducción.

El intérprete es un programa residente en la memoria central que lee las instrucciones en lenguaje de alto nivel, detecta los errores, los comunica y, si no hay errores, convierte las instrucciones a código interno y las ejecuta cuando se le indica.

Las principales diferencias entre un programa interpretado y compilado son las siguientes:

— La zona de memoria necesaria para operar con un intérprete es menor que la que se precisa para operar con un compilador.
— El programa compilado se ejecuta más rápidamente que el interpretado. Ello se debe a que el programa interpretado es ejecutado por otro programa que, a su vez, lo es por el ordenador.
— Es más fácil programar de forma interactiva contando con un intérprete, ya que avisa de los errores tan pronto como se cometen.

En la actualidad, la solución generalmente adoptada —sobre todo para lenguajes de gran difusión, como el BASIC y el PASCAL— consiste en ofrecer ambas posibilidades: intérprete y compilador. La primera opción facilita la programación, prueba y depuración de los programas. La segunda agiliza su ejecución en la fase de producción.

El primero fue el ALGY, creado en 1961, aunque también cabe citar el FORMAC, MATHLAB, ALTRAN, FLAP, MAGIC PAPER y SML.

f) *Lenguajes para manejo de ficheros y bancos de datos*

El incremento de la cantidad de información a manipular dentro de un proceso obligó a perfeccionar la gestión de los datos. Ante esta necesidad se empezaron a desarrollar lenguajes y sistemas para el tratamiento de ficheros y bancos de datos. Estos sistemas suelen integrarse bajo las siglas IMS (Information Management System), DMS (Data Management System), DBS (Data Base System), etc.

g) *Lenguajes especiales*

Esta categoría integra a los lenguajes que se utilizan en campos especializados y que, por tanto, no son de uso general. Se pueden agrupar por áreas de explicación; así, existen lenguajes para el control de máquinas herramientas (APT, AUTOSPOT, PRONTO, etc.), para ingeniería civil (COGO, STRESS, ICETRAN), diseño lógico (LOTIS, LDP), simulación (DYANA, DYNAMO, SIMULA, GPSS, SIMSCRIPT), diseño de compiladores (CLIP, TMG, META/5), análisis de microfotografías (BUGSYS), proceso y edición de textos (ES-1, SAFARI, IBM-DATATEXT), salidas gráficas (DIALOG, PENCIL, GRAF), desarrollos y estudios informáticos (lenguaje C), redes de datos y telecomunicación, y otras muchas aplicaciones.

La traducción

La traducción de un programa escrito en lenguaje de alto nivel la realiza otro programa, especializado en esta tarea, denominado *compilador.*

Durante el proceso de compilación (traducción por parte de un compilador) se comprueban los posibles errores sintácticos cometidos por el programador, así como la falta de definición de variables y otros errores, siempre que éstos no sean de organización lógica.

Al concluir el proceso se genera un listado final de errores detectados. Si finaliza el proceso sin error se puede obtener un listado del programa fuente, el listado del programa objeto, la tabla de variables y direcciones, la relación de subrutinas utilizadas, etc.



El empleo de programas de traducción de tipo «intérprete» permite al programador trabajar de forma interactiva, con el consiguiente ahorro de tiempo en la depuración y puesta a punto de los programas.

¿Cómo elegir un lenguaje?

La elección de un lenguaje depende de dos factores:

1. De la naturaleza del problema a resolver.
2. De que dispongamos para nuestro ordenador del compilador adecuado.

Aunque la confección de un programa consiste en crear una secuencia de instrucciones que realicen la toma de datos, ejecuten el algoritmo correspondiente e impriman el resultado, disponer de un lenguaje cuya estructura esté en consonancia con nuestras formulaciones lógicas hará que el trabajo sea mucho más fácil y eficaz.

Para saber más

¿Qué es un error sintáctico?

Los lenguajes de alto nivel, al igual que los lenguajes humanos, tienen una gramática específica. Las instrucciones, al igual que las frases de un lenguaje, deben construirse respetando unas reglas de sintaxis. Cuando no se obedecen estas reglas se está cometiendo un error sintáctico que el compilador es capaz de detectar.

¿Puede un compilador detectar un error lógico?

No puede, ya que el compilador no adivina el pensamiento del programador y, por tanto, es incapaz de detectar este tipo de error. Por ejemplo, no puede saber si donde está escrito A+B debería ser A—B o A×B.

¿Es siempre necesario compilar un programa escrito en lenguaje de alto nivel antes de ejecutarlo?

No siempre es necesario. A veces puede utilizarse un programa llamado *intérprete* que de alguna forma simultanea la traducción con la ejecución.

¿Qué es un lenguaje formal?

Se llaman lenguajes formales a aquellos cuyo diseño deriva de una gramática formal, con un alfabeto y unas leyes de deducción. Son de una gran importancia teórica, y uno de sus principales estudiosos fue Chomsky.

Cuando se usa un intérprete en lugar de un compilador, ¿se puede ejecutar varias veces un programa?

Cada vez que se quiera ejecutar es preciso recurrir al programa «intérprete», ya que el código objeto derivado del proceso de traducción y posterior ejecución línea a línea no se almacena en el ordenador.

Cuatro lenguajes evolucionados

Basic, Fortran, Cobol y RPG



Son muy pocos los usuarios de un sistema ordenador que no se han lanzado alguna vez a dialogar con la máquina en BASIC. Este popular lenguaje —sin duda alguna el más extendido en el universo informático— forma parte de los pioneros que han llevado a la ciencia informática al lugar de privilegio que hoy ocupa en la sociedad moderna. Compartiendo las posiciones de honor se encuentran también el FORTRAN y COBOL.

En los próximos apartados se dará un repaso a la naturaleza y funcionalidad de estos tres lenguajes y de un cuarto, el RPG, que también ha llegado a ostentar una gran relevancia empujado por su idoneidad para la programación de determinado tipo de aplicaciones en ordenadores IBM.

El lenguaje BASIC

El nombre BASIC está formado por las iniciales de *Beginner's All-Purpose Symbolic Instruction Code* (Código de instrucciones simbólicas de uso general para principiantes) y fue diseñado inicialmente para enseñar a programar.

No obstante, ha llegado a ser un lenguaje tan completo y con tantas características que se ha extendido su uso entre los programadores experimentados.

Fue desarrollado durante los años 1963 y 1964, en el Dartmouth College de Hannover (New Hampshire), bajo la dirección de los profesores Kemeny y Kurtz, recibiendo el apelativo de BASIC «de tiempo compartido». Otra versión, denominada BASIC «secuencial», se desarrolló en la Universidad de Washington, bajo la dirección de William F. Sharpe.

Desde entonces ha estado en continuo avance y ha habido numerosas implementaciones y dialectos, como, por ejemplo: BASIC extendido, BASIC avanzado, super BASIC, C-BASIC-II, BASIC-80, M-BASIC, etc. El estándar fue definido por ANSI en la norma X3.60-1978.

Aunque todos cumplen las mismas normas de base, cada desarrollo difiere de los demás. En estas páginas nos referiremos principalmente al MS-BASIC, desarrollado por Microsoft Corporation.

Modos de operación

El BASIC es fácil de aprender, y los programas escritos en este lenguaje pueden procesarse por medio de un intérprete que ejecuta directamente los programas en lugar de someterlos a una compilación previa.

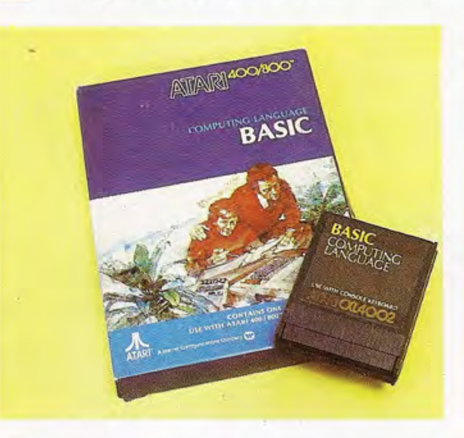
Son dos los modos de trabajo característicos de un intérprete BASIC: directo e indirecto.

En modo *directo* las instrucciones no se preceden de un número de línea y se van ejecutando según termina su introducción. Los resultados se presentan en la pantalla inmediatamente y se almacenan para poder usarlos en operaciones posteriores. Las instrucciones se pierden, por lo que en este modo se opera de forma semejante a como lo hace una calculadora.

En el modo *indirecto* es necesario in-



El BASIC es el lenguaje alto nivel que mayor popularidad y difusión alcanza en nuestros días. La facilidad de aprendizaje del BASIC está motivada en gran medida por la disponibilidad de programas «intérpretes» de alto caracter interactivo.



Al partir del BASIC original, se han creado múltiples dialectos, desde versiones simplificadas para microordenadores domésticos hasta potentes dialectos orientados a la programación de aplicaciones.



El nombre de BASIC está formado por las iniciales de «Beginners All-purpose Symbolic Instruction Code» (código de instrucciones simbólicas de uso general para principiantes); apelativo muy adecuado para su versión original.

Para saber más

¿Qué significa ANSI?

El término ANSI está formado por las iniciales de American National Standard Institute, que es el organismo que estandariza las características de los productos. Se subdivide en comités y subcomités que tratan de temas específicos, uno de los cuales trabaja en lenguajes de programación.

¿Qué es el «debugging»?

Es el gerundio del verbo «debug», que significa depurar. Se emplea este término para indicar que se depura un programa, es decir, que se eliminan los errores.

¿Qué son palabras claves o reservadas?

Son aquellas que, como los verbos del BASIC o las instrucciones del sistema operativo, son entendidas por el ordenador para realizar una función específica. Por ello no pueden ser usadas por el programador como nombre de variable.

¿Qué pasa cuando una constante, o el valor de una variable, definida como de simle precisión, tiene más cifras de las que permite la versión del lenguaje?

El número se trunca, es decir, pierde las cifras que sobrepasan la capacidad de almacenaje asignada.

¿Qué es un verbo en un lenguaje de programación?

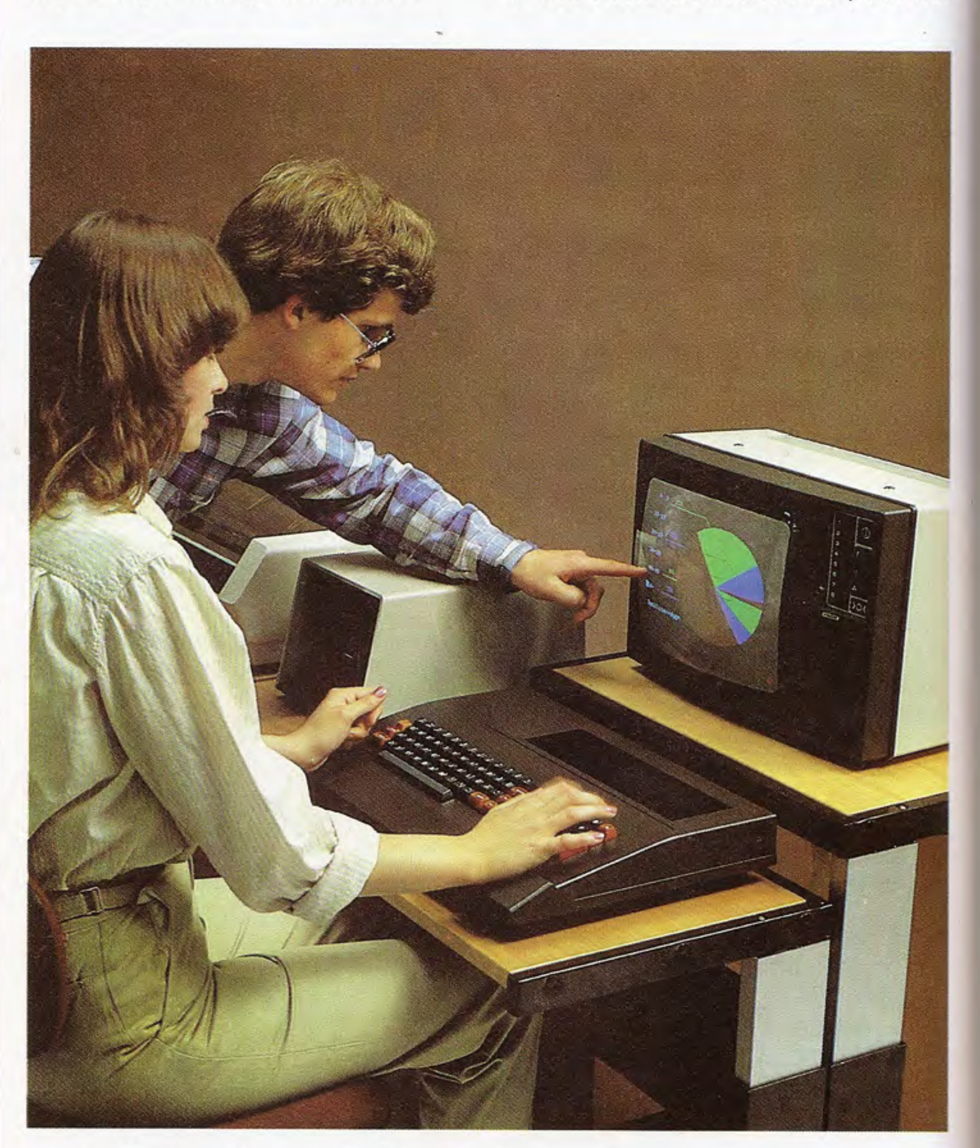
Es una palabra clave que indica la acción que debe ejecutar el ordenador.
Por ejemplo, PRINT es un verbo del BASIC que indica que se va a utilizar la pantalla para presentar resultados.

troducir el número de línea que identificará la secuencia de instrucciones dentro del programa. El programa almacenado en memoria puede ser ejecutado tantas veces como queramos sin más que introducir la instrucción RUN. El modo indirecto permite la depuración de programas. Además se puede pasar de un modo a otro automáticamente al detectarse errores de sintaxis, por lo que la labor, en unos casos de aprendizaje y en otros de «debugging» o puesta a punto, es muy sencilla para el usuario.

Elementos del lenguaje BASIC

Un programa escrito en BASIC está formado por un conjunto de líneas que contienen las instrucciones BASIC necesarias para llevar a cabo una tarea.

El formato de una línea es el siguiente:
[nnnnn] instrucción [: instrucción...]
siendo «nnnnn» el número de línea, que no es necesario si trabajamos en modo directo. Este puede ser cualquier número entero de uno a cinco dígitos y tiene un doble cometido: indicar en qué orden



El BASIC es el lenguaje de alto nivel más extendido en el campo de los microordenadores, debido a su fácil aprendizaje, la existencia de intérpretes y la gran diversidad de programas disponibles.

deben ejecutarse las instrucciones y proporcionar puntos de referencia para las instrucciones de salto. Los programas se ejecutan en orden de números de línea crecientes, con independencia del orden en el que se hayan introducido. Entre el número de línea y la instrucción debe haber un espacio en blanco.

La instrucción consta de dos partes:

— el *verbo,* que indica la acción a realizar;
— los *parámetros,* que completan la instrucción: especifican las variables a utilizar, la operación o función de cálculo, etc.

En una línea pueden escribirse una o varias instrucciones, con la condición de que vayan separadas por dos puntos (:).

La línea se termina cuando se acciona la tecla <return>.

Algunas versiones del lenguaje BASIC permiten que una línea de programa ocupe más de una línea de pantalla, mediante el empleo adecuado de las teclas de control.

En un programa en MS-BASIC se pueden usar caracteres alfabético, numéricos y especiales:

— Los alfabéticos son las letras del alfabeto, tanto mayúsculas como minúsculas.
— Los numéricos son los dígitos del 0 al 9.
— Los especiales incluyen tanto caracteres visibles (+, —, %, etc.), como de control (<carriage return>, fin de línea; <line feed>, sigue la línea en la próxima panta-



El impulso definitivo del lenguaje BASIC ha llegado de la mano de los microordenadores, una de cuyas familias es la de los microordenadores familiares. Estos equipos están difundiendo el uso de este lenguaje incluso a nivel doméstico.

lla, etc.) e incluso invisibles (tono audible o timbre en el terminal).

Constantes

Las constantes son los valores fijos que el BASIC utiliza durante la ejecución. Hay constantes de dos tipos: alfanuméricas y numéricas:

- Las *constantes alfanuméricas* son cadenas de carateres que tienen entre 0 y 255 caracteres alfanuméricos encerrados entre comillas.

Ejemplos: «PEPE», «2000.00 pts.», «Nombre del empleado»...

- Las *constantes numéricas* son números positivos o negativos que no pueden contener comas, sólo en determinados casos el punto decimal. Hay cinco tipos de constantes numéricas:

— *Enteras:* Números enteros. No tienen punto decimal.

— *De coma fija:* Números reales positivos o negativos. Deben presentar el punto decimal.

— *De coma flotante:* Números reales positivos o negativos representados en notación científica, esto es, en notación exponencial. Constan de una mantisa (constante en coma fija, con o sin signo), seguida por la letra E (o D en el caso de doble precisión) y por un entero con o sin signo (el exponente al que hay que elevar la base 10 para obtener el número deseado).

Ejemplos:

7.0378E - 05 = $7.0378 \cdot 10^{-5}$ = = 0.000070378.

4.5E + 02 = $4.5.10^{2}$ = 450.

— *Hexadecimales:* Números hexadecimales con el prefijo &H.
Ejemplos: &H76, &HA30B.

— *Octales:* Números octales con el prefijo &O.
Ejemplos: &O345, &O731.

En muchas versiones del BASIC, las constantes numéricas pueden ser de simple o de doble precisión. Las de doble precisión son útiles para cálculos que necesitan emplear muchas cifras exactas, de ahí que su almacenamiento necesite más bytes.

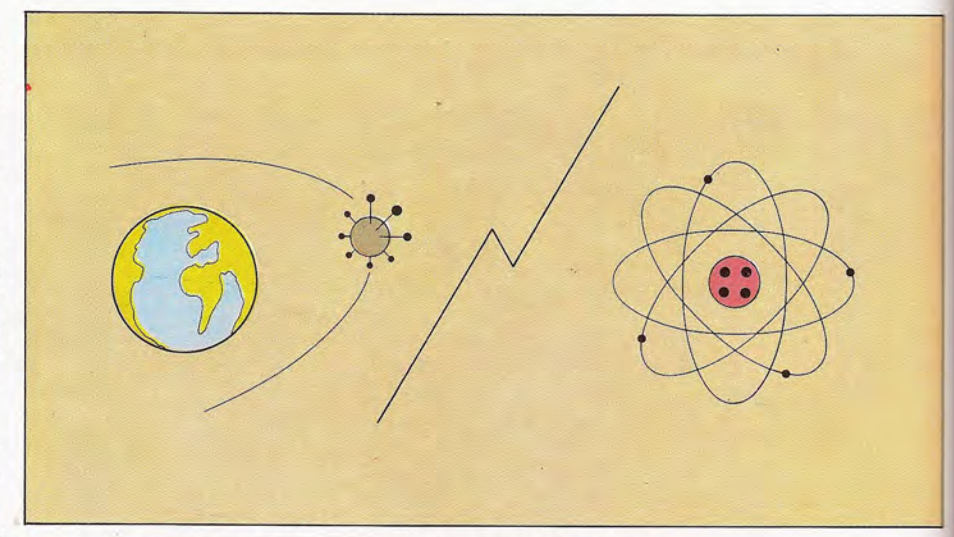
Variables

Una variable es un área de memoria a la que se asigna un nombre y en la que se almacenan datos. El nombre de la variable es la dirección simbólica que el intérprete convertirá en la dirección real del dato asignado a la variable.

El nombre de una variable lo crea el programador y no puede coincidir con ninguna palabra utilizada por el BASIC para designar comandos o funciones.

Un nombre válido está constituido por una letra mayúscula seguida de cualquier cantidad de letras o números (el límite en longitud depende de las características del intérprete BASIC que se utilice).

Es muy importante reseñar que en mu-



El lenguaje FORTRAN está orientado especialmente a aplicaciones de tipo científico y técnico, desde la determinación de las órbitas de los satélites espaciales, a la definición de distancias entre elementos atómicos.



El primer manual de FORTRAN fue publicado por IBM en 1956, año en que comenzaron a redactarse numerosas versiones o dialectos, hasta llegar a nuestros días en los que el FORTRAN ya ha sido superado en eficacia por otros lenguajes.

chas versiones sólo son significativos los dos primeros caracteres, por lo que, por ejemplo, las variables CU, CUEN y CUENTA representarían la misma dirección de memoria.

Hay diversos tipos de variables que se distinguen por el carácter incluido al final de su expresión:

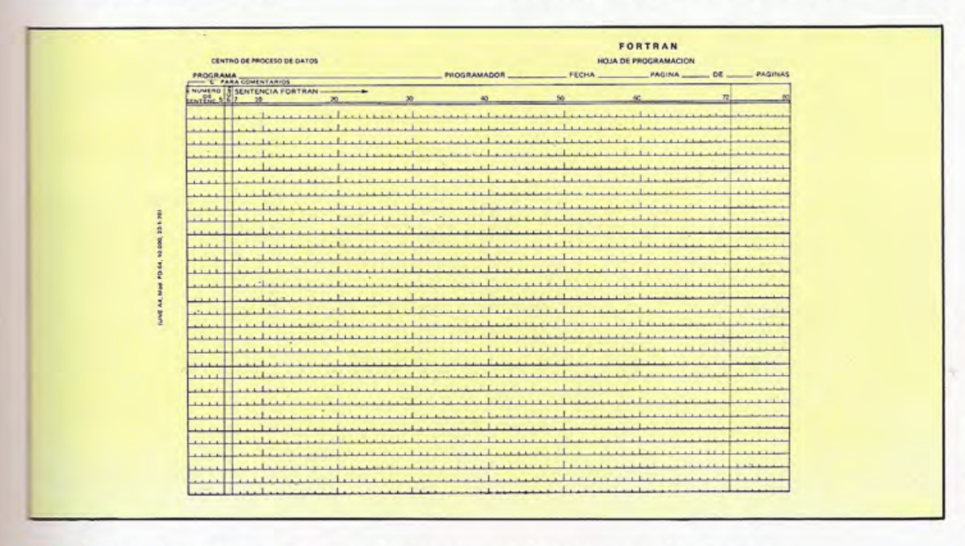
— Las variables de *cadena de caracteres* acaban en $.
— Las variables *enteras* acaban en %.
— Las de *simple precisión* en !.
— Las de *doble precisión* en ≠.

Las variables de diverso tipo pueden utilizar las mismas letras para su presentación; así por ejemplo, A% y A$ son dos variables distintas.

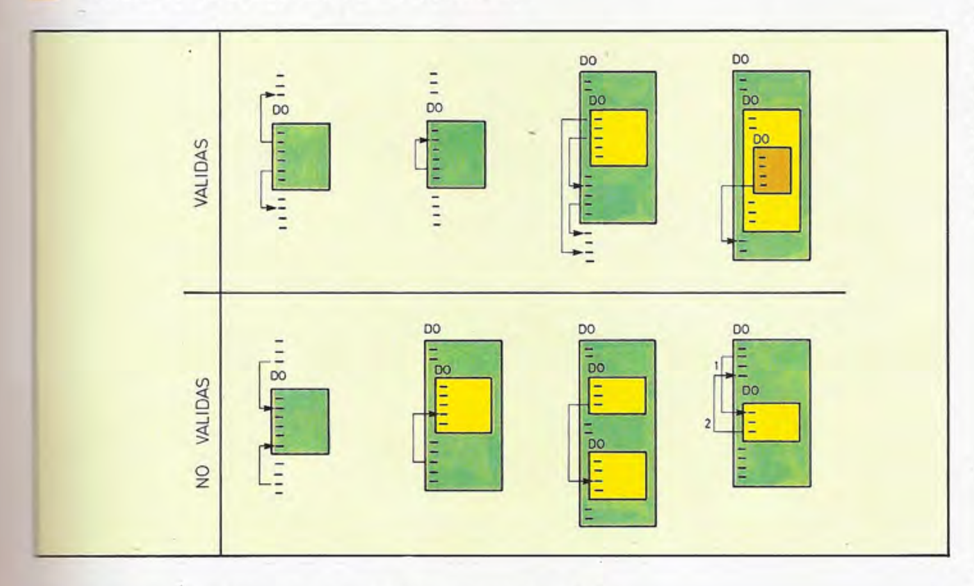
Por último, nos ocuparemos de las *matrices* o tablas de valores referenciados con una denominación común. Cada elemento de la matriz se diferencia de los demás por la posición que ocupa dentro de la tabla; al efecto se utilizan subíndices. Los subíndices no pueden ser ni cero ni negativos.

Ejemplos: A$(7) sería el séptimo elemento de una tabla de cadenas de caracteres de una sola dimensión que, por ejemplo, podría estar constituida por los meses del año.

B!(3,4) sería el elemento que ocupa la fila 3 de la columna 4 en una matriz de datos numéricos de precisión simple.



La figura muestra una hoja de programación típica empleada para la redacción de programas en lenguaje FORTRAN. Dispone de 80 columnas. Las ocho últimas se usan para identificar el programa y el número de secuencia dentro del mismo.



Las instrucciones de tipo "DO" para el control de bucles pueden anidarse, aunque no deben interferirse. El cuadro muestra las estructuras válidas y erróneas de este tipo de instrucciones.

El lenguaje FORTRAN

Desde la aparición del ordenador electrónico, las universidades y centros de investigación técnica y científica comprendieron la gran ayuda que éste les iba a proporcionar. Pero existía la barrera de los lenguajes de programación. Los lenguajes de máquina y de ensamble estaban bastante lejos de la forma de expresión técnica (modelos matemáticos, expresiones aritméticas, ecuaciones, etc.). De ahí surgió la idea de un lenguaje para la resolución de problemas científicos mediante técnicas de cálculo numérico. El primero de estos lenguajes fue el SHORT CODE, creado en 1949 por el doctor Mendy para UNIVAC. Unos años más tarde, en 1953, aparece el SPEEDCODING, de Backus para IBM.

Treinta años de FORTRAN

El mismo Backus inició el desarrollo del lenguaje FORTRAN en noviembre de 1954, publicando IBM el primer manual en 1956.

El FORTRAN II aparece en junio de 1958, aportando importantes amplia-

ciones fundamentalmente en el campo de las subrutinas, y el FORTRAN IV lo hace en 1962. A partir de entonces fueron varias las firmas que contribuyeron a su expansión, de tal forma que existen en la actualidad más de 50 compiladores.

Para equipos pequeños existen diversas versiones, siendo de destacar el FORTRAN-80 de Microsoft (firma especializada en software para ordenadores personales y micros). Esta contiene 28 tipos de instrucciones diferentes, entre ellas todas las normalizadas por ANSI. Las principales ventajas del FORTRAN son su capacidad para manejar expresiones matemáticas y su velocidad de cálculo, por lo que es apropiado para las aplicaciones científicas en las que lo importante es el volumen de cálculo y no la entrada/salida.

Trabaja sólo con compilador, realizando la compilación en un solo paso. Debido a su antigüedad, y a pesar del gran número de revisiones que ha sufrido, hoy día es bastante inferior a otros lenguajes, tales como el Pascal o versiones avanzadas del BASIC. De todas formas, los programas escritos en FORTRAN constituyen todavía un elevado porcentaje de las bibliotecas de programas técnicos y científicos.

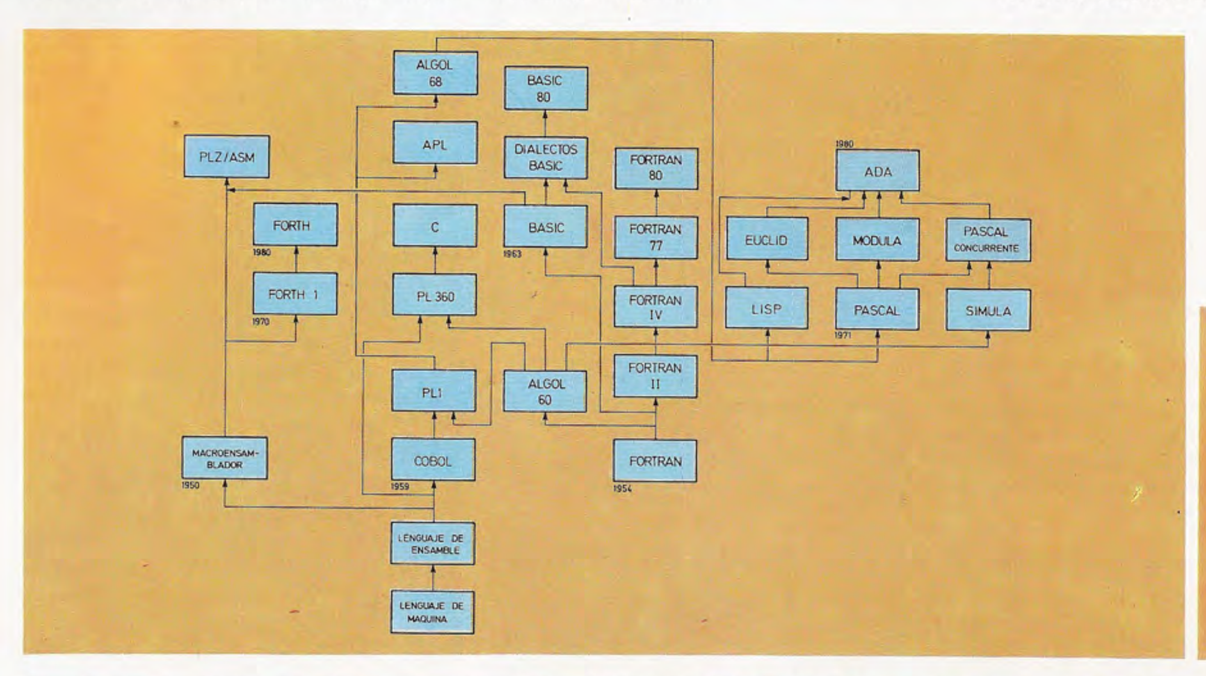
Las hojas de programación

El FORTRAN utiliza los caracteres alfabéticos de la A a la Z, los dígitos del 0 al 9 y los caracteres especiales siguientes:

(espacio) = + – * / () , . $ '

Con ellos se escriben los programas en una hoja de diseño especial que admite 80 caracteres por línea.

Si en la columna 1 aparece una C, in-



El gráfico muestra la evolución histórica de los lenguajes de programación. Puede observarse cómo el BASIC tomó en un principio conceptos y sentencias del FORTRAN.



Hoy en día es aún muy extensa la biblioteca FORTRAN de aplicaciones científico/técnicas, adaptadas a los entornos más dispares.

dica que es un comentario que no será traducido por el compilador.

Una cadena de 1 a 5 dígitos, ajustados a la derecha, sirve de etiqueta para referenciar una sentencia. Por consiguiente, sólo se etiquetarán aquellas instrucciones que sea preciso identificar.

La sentencia o instrucción se escribe desde la columna 7 a la 72. Algunas versiones permiten continuar la sentencia en otra línea, poniendo en la línea siguiente un carácter distinto de 0 o un espacio en la columna 6. Las posiciones 73 a 80 no son tenidas en cuenta por el compilador; se usan para identificar el programa y el número de secuencia dentro del mismo.

Datos y operaciones

El FORTRAN admite constantes y variables enteras, reales y de doble precisión. En general las de doble precisión, tan necesarias en el cálculo técnico-científico, permiten hasta 16 cifras decimales.

También se admiten constantes Hollerith, es decir, cadenas de caracteres que se escriben entre apóstrofes (por ejemplo, 'NOTA').

Las variables emplean nombres simbólicos que tienen una letra como primer carácter. El resto, hasta 8, pueden ser letras o números. Si no se usan sentencias de especificación, el compilador entiende que toda variable cuya primera letra es I, J, K, L, M o N es una variable entera, mientras que las que empiezan por letras desde la A a la H y desde la O a la Z son variables reales.

Las variables matriciales se representan con el nombre seguido por los subíndices colocados entre paréntesis y separados por comas. Así A(3,1,4) representa un elemento de una matriz tridimensional.

Las operaciones permitidas son: exponenciación (**), multiplicación (*), divi-



El lenguaje COBOL nació en 1959, orientado específicamente a la gestión administrativa y a los negocios. El objetivo de sus creadores era convertirlo en un lenguaje universar para todos los ordenadores.

Para saber más

¿Qué significa FORTRAN?

La palabra FORTRAN está formada por las sílabas iniciales de FORmula TRANSlation, cuyo significado, traducción de fórmula, es revelador de su utilidad para problemas técnicos y científicos.

¿Por qué es necesario el cálculo numérico?

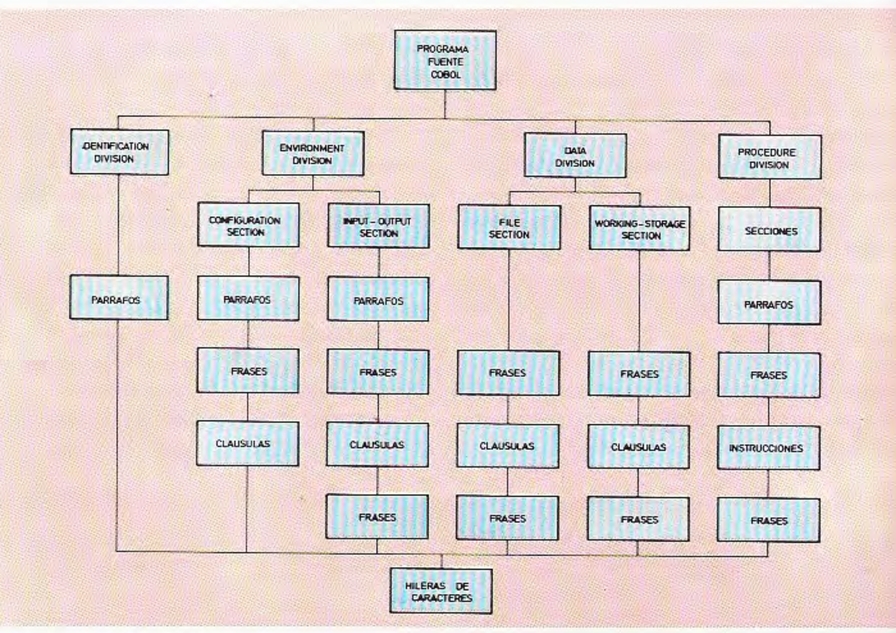
El lenguaje FORTRAN no admite más operadores que los aritméticos, mientras que los problemas técnicos precisan trabajar con integrales, derivadas, ecuaciones diferenciales, interpolaciones, ajustes de curvas, etc. El cálculo numérico es una colección de técnicas que permiten sustituir estos cálculos por conjuntos de operaciones aritméticas que dan una solución lo suficientemente aproximada.

¿Son suficientes dieciséis decimales para los problemas científicos?

Muchas veces no, ya que al sustituir los cálculos reales por métodos aproximados de cálculo numérico pueden ser precisas más cifras. El programador debe conocer la teoría de errores y tenerlo en cuenta a la hora de programar.

¿Por qué una línea FORTRAN sólo admite 80 caracteres?

Es una reminiscencia de las tarjetas perforadas, ya que cuando se desarrolló el FORTRAN éstas constituían el medio de entrada más ampliamente utilizado.



La misión de las divisiones en las que se estructura el lenguaje COBOL es informar al ordenador de aspectos parciales del programa

HOJA DE CODIFICACION COBOL

Programa ______ Preparado por ______

Fecha ______ Pág. __ de __

Número de Secuencia (Pág. / Línea)	TEXTO	Identificación del Programa

La figura muestra la organización de una hoja de codificación COBOL. Obsérvese la gran similitud que guarda con las hojas de programación en lenguaje FORTRAN.



En la actualidad existen numerosas versiones o dialectos del COBOL 60 original. Cada uno de ellos supone una mejora o adaptación a necesidades del usuario, pero siempre dentro de la orientación primera: la gestión administrativa y los negocios.



En lenguaje COBOL (Common Business Languaje) pueden encontrarse miles de aplicaciones orientadas a la gestión administrativa, así como en los más diversos formatos para su entrada en el ordenador: disquete, casete, etc.

sión (/), suma (+) y sustracción (—). Las normas de prioridad de operaciones y uso de paréntesis propias del BASIC son también de aplicación en el FORTRAN.

Existen numerosas funciones que permiten cálculos trigonométricos, logarítmicos, etc. El programador puede crear además sus propias funciones.

Instrucciones

Veremos a continuación algunas de las instrucciones más características del FORTRAN:

- *Instrucciones de asignación.* Tienen el formato

variable = expresión

y sirven para dar a una variable el valor obtenido en el cálculo de una expresión aritmética.

Ejemplo:

Z(J) = (—B—SQRT(B*B—4 *A *C))/(2 *A)

que atribuye al elemento J de la serie Z el valor de una raíz de la ecuación $AX^2 + BX + C = 0$.

- *Instrucciones de control.* Alteran la secuencia normal de ejecución.

Salto incondicional: GO TO n (siendo n una etiqueta).

Salto calculado: GO TO (n_1, n_2,..., n_n), expresión aritmética (continúa en n_1, n_2, etc., según que el valor de la expresión sea 1, 2, ...).

Salto condicional: IF(exp) n_1, n_2, n_3 (va a n_1, n_2 o n_3, según que el valor de la expresión sea negativo, cero o positivo).

Bucle: DO n_1 ind = exp_1, exp_2, exp_3 (repite todas las instrucciones que siguen hasta la n_1 variando el valor del índice desde exp_1 a exp_2 en incrementos de exp_3 en exp_3).

Fin de bucle: CONTINUE (se usa como última instrucción de un DO, en lugar de la que correspondería en caso de transferencia).

Otras: PAUSE, STOP, END.

Para saber más

¿Qué significa COBOL?

La palabra COBOL está formada por las iniciales de COmmon Business Oriented Language (Lenguaje Común Orientado a los Negocios).

¿Qué es el CODASYL?

CODASYL son las siglas de Conference on Data Systems Languages (Conferencia sobre Lenguajes de Sistemas de Datos). Este comité fue convocado por el Departamento de Defensa de los Estados Unidos y estaba formado, entre otros, por las empresas Burroughs, IBM, Honeywell, RCA, Remington, Sperry Rand, Sylvania, USAF, Navy y el National Bureau of Standards.

¿Qué significa la página y línea en el número de secuencia de la hoja de programación COBOL?

Sólo un número de orden, aunque no tiene que estar consecutivo. El criterio general es numerar las líneas de 10 en 10 para permitir la intercalación de nuevas sentencias. La estructura sigue recordando la época en que se trabajaba con fichas perforadas (80 columnas) y la posibilidad de clasificar las fichas mediante el número de secuencia, utilizando una clasificadora.

¿Qué se entiende por verbo COBOL?

Un verbo o palabra COBOL es una hilera de caracteres que reconoce el compilador y que permiten generar el programa objeto. Los verbos son palabras reservadas del lenguaje.

- *Instrucciones de declaración:* Para reservar zonas de memoria.

Dimensionado de matrices: DIMENSION nombre matriz (i_1, i_2, ..., i_n).

Para compartir la misma posición de memoria: EQUIVALENCE (V_1, V_2, ... V_n).

Para establecer correspondencia entre variables de diferentes unidades del programa = COMMON V_1, V_2, ..., V_n.

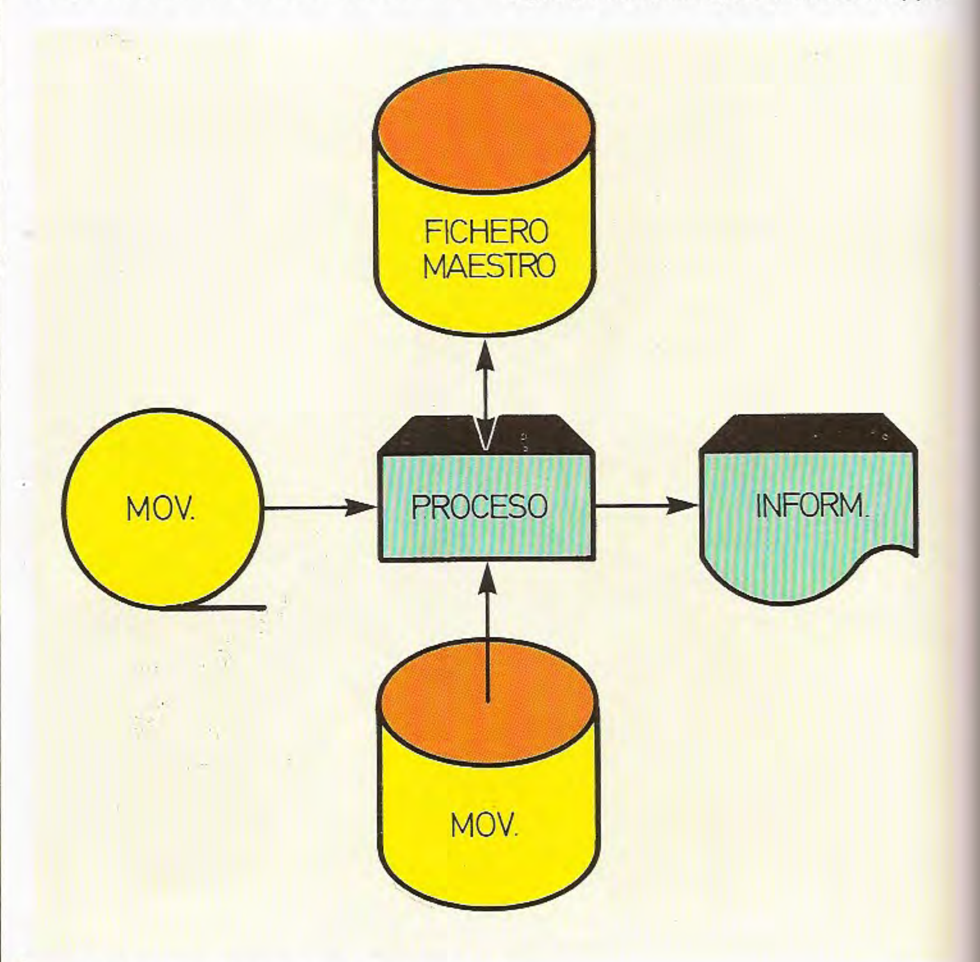
Otras: INTEGER, REAL, DATA.

- *Instrucciones de entrada/salida.* Todas las entradas se realizan con la instrucción READ, mientras que las salidas se ejecutan mediante la instrucción WRITE. Ambas llevan asociadas una instrucción FORMAT que define el formato de lectura o escritura.

En resumen, el FORTRAN es un lenguaje muy fácil de aprender para las personas acostumbradas a la notación matemática, aunque hoy día está siendo superado por otros lenguajes más modernos.

El lenguaje COBOL

Aunque más lentamente que en el campo científico, los ordenadores entraron en el área de gestión con gran fuerza, sobre todo debido a la aparición de lenguajes de alto nivel orientados específicamente a los negocios. El primer lenguaje de este tipo, históricamente hablando, fue el FLOW-MATIC, que en 1955 estableció el concepto de lenguajes de programación basados en palabras del lenguaje natural (en este caso el inglés). Fue creado por el doctor Hopper



El objetivo esencial del RPGII es facilitar la obtención de datos, manipularlos y crear informes de salida.

para UNIVAC. No obstante, el lenguaje de gestión que alcanzó más rápidamente la popularidad fue el COBOL, desarrollado a partir de 1959 por CODASYL. Conocido en un principio por COBOL 60, pretendía ser un lenguaje común a todos los ordenadores. Posteriormente han surgido nuevas versiones, por ejemplo: el COBOL ANSI 74, el COBOL-80 de Microsoft, el CIS-COBOL (que facilita el manejo de pantallas) y el RM-COBOL (para microprocesadores).

Ventajas e inconvenientes

El lenguaje COBOL tuvo un gran éxito, ya que incluía un concepto básico: el diseño de los datos es independiente de los algoritmos que van a operar con ellos. Este principio permite la definición minuciosa de los elementos a utilizar. Por lo demás, este lenguaje tiene gran capacidad de manejo de campos alfanuméricos, lo que es útil para programar salidas impresas en sistemas de gestión.

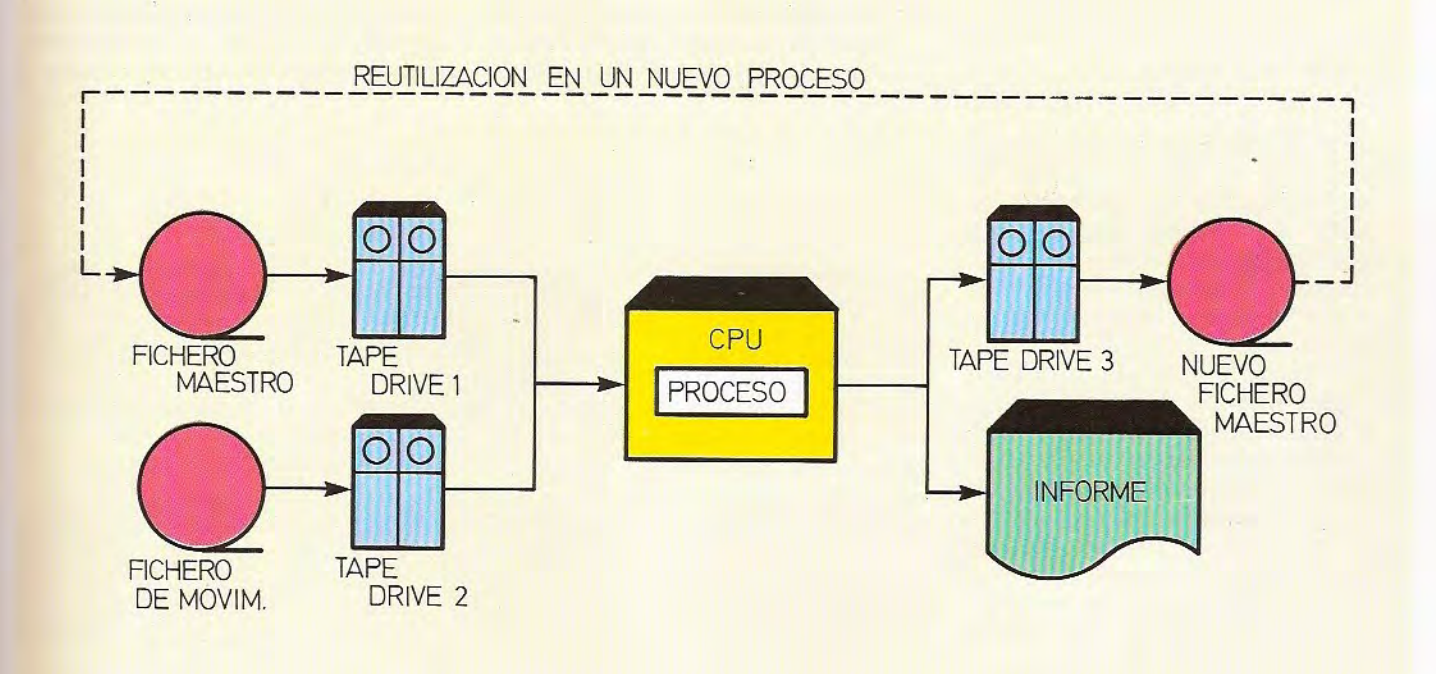
A pesar de haber sido superado por otros lenguajes, aún se utiliza ampliamente. Esto se debe a las numerosas aplicaciones desarrolladas a lo largo de veinte años y a la experiencia acumulada por los programadores.

Inicialmente era un lenguaje batch, si bien hoy existen numerosas versiones interactivas que incorporan instrucciones de acceso a pantalla.

Es el lenguaje más estándar de los existentes en la actualidad y los programas escritos en COBOL se pueden implementar fácilmente en distintos ordenadores. Además, su manejo no hace necesario un conocimiento profundo de matemáticas.

Estructura general

Un programa en lenguaje COBOL se encuentra jerarquizado en la siguiente forma: Division, Section, Paragraph, Sentence, Statement, Word y Character. Las divisiones son cuatro, cada una de las cuales informa al compilador de un aspecto del programa. Las divisiones



Esta es la representación de un proceso de tipo BATCH cuando los dispositivos de acceso utilizados son de tipo secuencial.

deben escribirse exactamente en el mismo orden en que se reseñan:

- *Identification Division:* Identifica al programa. Además del nombre del mismo, incluye información adicional sobre el autor, fecha, etc.
- *Environment Division:* Adapta el resto del programa a la configuración del sistema y a su sistema operativo. En teoría sería la única división que habría que cambiar si se traslada el programa a otro ordenador.
- *Data Division:* Describe las estructuras de los datos que van a ser procesados.
- *Procedure Division:* Contiene las instrucciones con las que el ordenador procesará los datos; esto es, el programa propiamente dicho.

Tipos de sentencias

Las divisiones se estructuran en secciones, según se indica en el gráfico adjunto. Mientras que los nombres de las secciones de la Environment Division y la Data Division están fijados por el propio lenguaje, en la Procedure Division los elige el programador, seguidos de la palabra «Section».

La Configuration Section describe el ordenador en el que se compila y ejecuta el programa; mientras que la Input-Output Section señala los periféricos utilizados y su asignación a los ficheros.

Las secciones de la Data Division son: File Section (para los ficheros), Working Storage Section (para las zonas de maniobra), Constant Section (para las zonas de constantes) y Report Section (para la descripción de salidas).

Los párrafos o subdivisiones de las secciones constan de una o más frases que tienen una función común. Los párrafos de la «Procedure» pueden ser fijados por el programador, mientras que los nombres de los párrafos de las otras instrucciones están fijados por el lenguaje.

Los párrafos de la Environment Division y Data Division están formados por frases (entries) y cláusulas, mientras que los de la Procedure Division lo están por frases e instrucciones.

Una instrucción es una combinación sintácticamente válida de palabras y símbolos, que comienza con un verbo COBOL. Existen tres tipos de instrucciones: del compilador, condicionales e imperativas.

La hoja de codificación

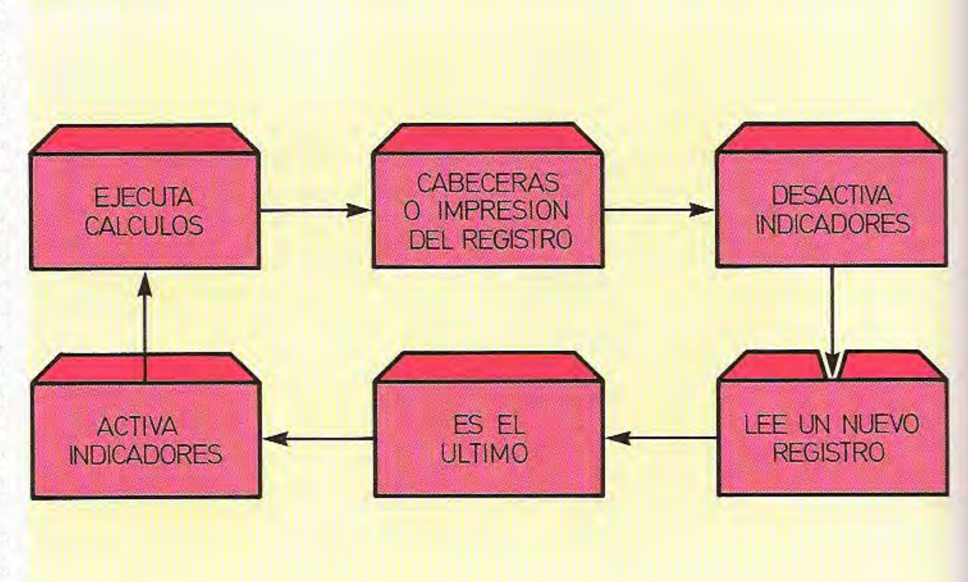
Los programas fuente en lenguaje COBOL se escriben en hojas de codificación normalizadas que recuerdan a las del FORTRAN. En este caso la línea de continuación se indica en la columna 7, mediante un guión, y la de comentario, mediante un asterisco en dicha columna 7.

Al codificar un programa en COBOL, o en cualquier otro lenguaje, deben seguirse unas ciertas reglas, cuya misión es disminuir las posibilidades de error. Así, por ejemplo, se debe utilizar lápiz suave y goma de borrar en lugar de bolígrafo; se debe poner un solo carácter en cada posición; escribir ciertos caracteres (tales como el cero y la O, la zeta y el 2, la I o el 1) de forma que se eviten errores.

El lenguaje RPG

RPG es la abreviatura de Report Program Generator (Programa Generador de Informes), que da nombre a un lenguaje de alto nivel relativamente fácil de aprender. Por su estructura, el programador no necesita conocer en profundidad la máquina, confiando esta tarea al compilador.

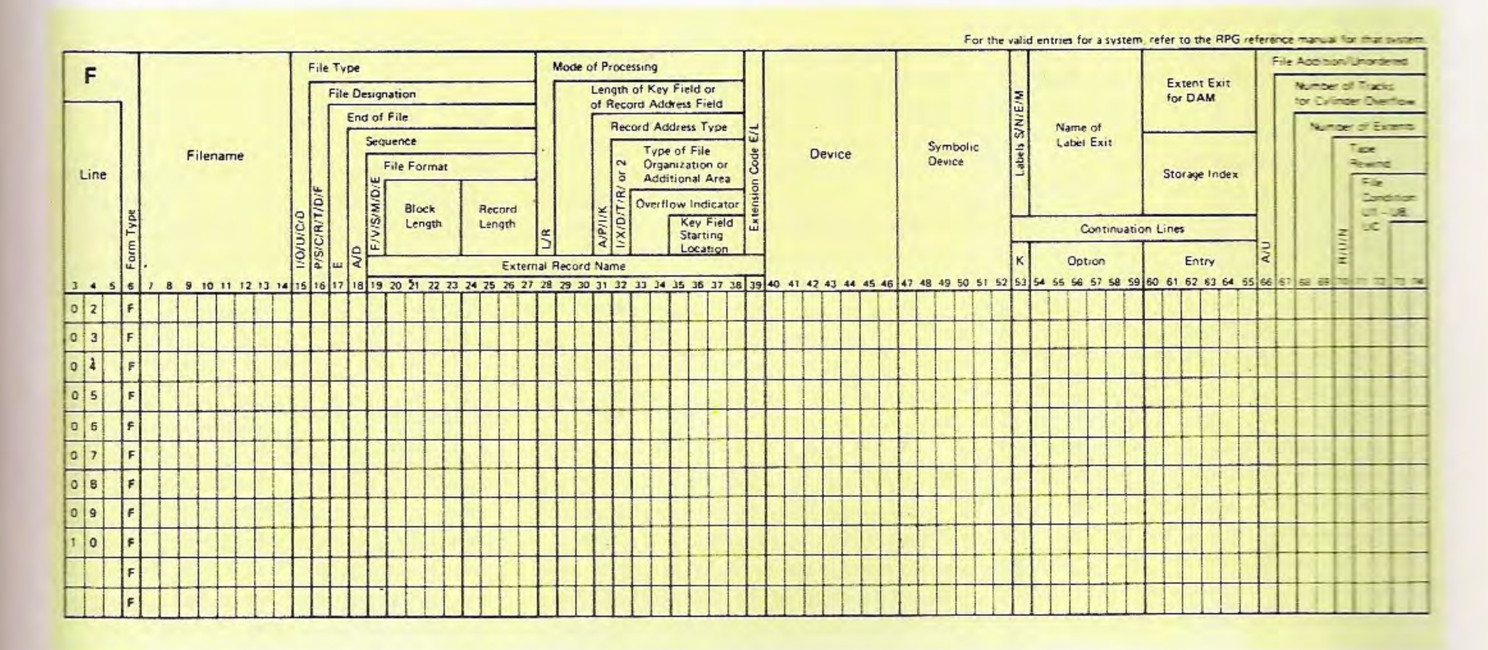
El RPG fue diseñado en los años sesenta por IBM para usarlo en sus propios sistemas, pasando a ser posteriormente un estándar en otras máquinas y



El ciclo lógico de programación en RPGII tiene lugar tal como se indica en la figura. Como puede verse, el desarrollo de este ciclo depende poco del programador y de sus necesidades, y viene fijado por la propia estructura del lenguaje.

Evolución de los lenguajes de programación

El primer lenguaje que se utilizó en la programación fue el lenguaje de *máquina,* codificado en binario o hexadecimal y que sólo era comprensible para la máquina. Debido a su complejidad para el programador se desarrolló el lenguaje de *Ensamble,* que aunque seguía estando próximo a la máquina, sustituía el código máquina por códigos nemotécnicos y simbólicos.
A mediados de los años cincuenta aparecen los lenguajes *Macroensambladores,* con potentes instrucciones para sustituir a los procesos de codificación nemotécnicos largos e incómodos.
Siguiendo los principios de los *Macroensambladores,* a principios de los setenta se desarrolló el FORTH 1, que dio lugar en los ochenta al FORTH utilizado en microprocesadores, así como al PLZ/ASM.
De forma paralela a los lenguajes *Macroensambladores* se desarrollaron lenguajes que se alejaban de la máquina y se aproximaban mucho más al problema. Los dos lenguajes históricamente más importantes son el FORTRAN y el COBOL, el primero dedicado al campo científico y el segundo ligado al campo comercial y de gestión.
El FORTRAN fue presentado por IBM en 1954, su desarrollo dio lugar en 1970 al FORTRAN IV y en 1977 al FORTRAN 77. En 1965 nace un lenguaje derivado del FORTRAN y que las universidades americanas empezaron a utilizar como lenguaje científico, el BASIC, que se ha desarrollado actualmente de tal manera que hoy es el lenguaje tradicional en los microordenadores. Otro lenguaje científico interesante que nació en el año 58 fue el ALGOL, cuya versión ALGOL 60 es la más representativa.
El COBOL se desarrolló en 1959, teniendo versiones mejoradas en los años 74 y 80, llamadas COBOL 74 y COBOL 80. A la vez que se desarrollaban estos tipos de lenguajes, al principio de los años sesenta se comienzan a diseñar lenguajes polivalentes, es decir, lenguajes que sirven para solucionar tanto problemas científicos como de gestión. Nace el PL/1, derivado del COBOL y del ALGOL 60; del PL/1 se deriva el APL, en los años setenta, que se utiliza en trabajos interactivos y en la enseñanza asistida por ordenador. No hay que pasar por alto que actualmente el BASIC es también un lenguaje polivalente.
Siguiendo con la idea de llegar a un lenguaje universal, se desarrolló en los años setenta un lenguaje derivado del ALGOL 60 y del ALGOL 68, llamado PASCAL, cuyas versiones (PASCAL UCSD) y dialectos afines son utilizados ampliamente.
En la década actual los lenguajes PASCAL RECURRENTE y LIPS (derivado del ALGOL 68) han conducido al lenguaje ADA, que en principio tiene un amplio futuro. Por ejemplo, el Departamento de Defensa de los Estados Unidos ha decidido que a partir de 1985, y con el fin de estandarizar sus aplicaciones, no aceptará ningún trabajo que no esté programado en ADA.



En la figura puede observarse la cabecera de una hoja de programación tipo F que corresponde a la descripción de las características de cada uno de los ficheros.

constructores. En la actualidad se utiliza la versión RPGII, que fue desarrollada en 1974.

RPGII

El objetivo básico de este lenguaje es permitir, sobre todo, extraer información y manipularla para crear informes escritos. Esto implica que es un lenguaje orientado preferentemente hacia la resolución de tareas de gestión. El RPGII carece de sentencias y de filosofía para resolver problemas técnicos y programación de sistemas, existiendo para ello otros lenguajes como pueden ser el FORTRAN o el «C». El RPGII está básicamente orientado a sistemas pequeños y medianos.

En BASIC las instrucciones de un programa se introducen en la máquina a modo de listado representativo del programa. Por el contrario, en RPGII lo que se hace es rellenar unas hojas tabuladas que le indican a la máquina una serie de variaciones sobre el ciclo interno; es decir, no se hace un programa como en BASIC.

En el tiempo de ejecución se lee un registro del fichero primario, se procesa según las indicaciones dadas y a continuación se imprimen los resultados. Teniendo escrito el programa en las hojas de codificación, el paso siguiente es pasárselo al compilador para que lo procese. Para ello, según el sistema, existen varios métodos. Uno de ellos se basa en pantallas formateadas de forma similar a las hojas de especificaciones RPGII, en las cuales se va introduciendo el programa fuente. Cada campo de dichas hojas tiene un significado concreto para el compilador. Podemos decir que los contenidos de estos campos pueden simbolizar «parámetros» para ejecutar acciones en consecuencia. Esta idea se asemeja a la filosofía de los programas de explotación de ficheros o bases de datos.

Básicamente, en un proceso de datos existe lo que se llama un *fichero maestro,* en el cual están almacenados una serie de datos que son acumulativos y descriptivos. Así, puede existir un *maestro* que tenga el número de la seguridad social, la cantidad total retenida hasta el momento, descuentos, etc.

Otro fichero suele tener los cambios o acciones realizadas durante un determinado período de tiempo. Constaría, por ejemplo, de pagos de nómina, detalle de las retenciones, etc. Este archivo se llamaría *fichero de movimientos.*

El programa o la cadena de programas tendría que tomar el *fichero de movimientos* y actualizar el *fichero maestro.* De esta forma se tiene toda la información actualizada. Otra acción fundamental, aparte de la actualización, sería presentar una serie de informes escritos que relacionaran, por ejemplo, todos los cambios efectuados en el *fichero maestro;* también tendría que indicar los errores encontrados, etc.

Teniendo en cuenta la descripción anterior, se puede imaginar el funcionamiento de un programa en RPGII como una secuencia constituida por los siguientes pasos: entrada proveniente de un fichero de trabajo, elaboración, cálculo y selección de la información leída, proceso de grabación en otro fichero de información ya procesada y, por último, realización de un listado que refleje la situación de ambos ficheros o los resultados intermedios.

Hojas de programación en RPGII

Para el lenguaje RPGII, cada una de estas cuatro acciones se describe en una hoja que contendrá toda la información necesaria para su ejecución.

Una primera hoja tendrá la descripción



Codificación y control de errores

La codificación consiste en establecer una correspondencia entre la información que queremos representar y su representación, de forma que a cada información le corresponde una y sólo una forma de representación.
Como el ordenador maneja sólo información binaria, toda la información, tanto numérica como alfabética, debe representarse mediante cadenas de bits. Se han utilizado multitud de sistemas de codificación de la información, tanto dentro como fuera del ordenador. Veamos algunas de las razones que han servido de base para construir los códigos más usuales:

- *Número de símbolos a representar*

Al tener que codificar las 26 letras, los 10 dígitos y unos 30 símbolos especiales, se necesitan por lo menos 60 configuraciones binarias, por lo que el número mínimo de elementos del código ha de ser 6 bits ($2^6=64$ combinaciones posibles). En general se usan 7 u 8 bits por la razón que se explica seguidamente.

- *Detección y corrección de errores*

La necesidad de que no se produzcan errores en las distintas etapas obliga a introducir mecanismos que detecten y corrijan los errores de forma automática. Esto se consigue utilizando más bits de los necesarios, los llamados bits de paridad. Se dice que un código es óptimo cuando para representar un símbolo se usa el menor número de posiciones binarias posibles. Cuando se emplean más de las estrictamente necesarias se dice que el código es redundante. Es esta redundancia la que asegura la fiabilidad del dódigo.

- *Rendimiento de un código*

Se define el rendimiento de un código como el cociente entre el número de informaciones codificadas y la cantidad total que podrían representarse con el código utilizado. Se da en tanto por ciento.
Así, por ejemplo, si empleamos 6 bits para representar sólo diez dígitos, el rendimiento será:

$$n = 100 \times \frac{10}{2^6} = \frac{1.000}{64} \simeq 15,6\ \%$$

de las características de cada uno de los ficheros que intervienen a lo largo del programa, así como cualquier otra indicación adicional al sistema. Por ejemplo, el nombre del fichero, tipo del que se trata, longitud del registro, del bloque, etc. Como esta hoja trata de «ficheros» (Files), se le suele llamar «hoja F» (todas las hojas llevan para su identificación en la columna 6 el tipo del que se trata; en nuestro caso estarán impresas con la letra «F»).

La «hoja I» (I de Input) contiene la descripción de los formatos de los archivos que se indican en la «hoja F». Se incluyen aquí, por ejemplo, los nombres de los campos, la posición en la que comienzan y en la que acaban, los indicadores de registro, etc.

Siguiendo con nuestra «simulación» de un proceso, ahora habrá que indicar las operaciones que queremos que el ordenador efectúe con los campos definidos anteriormente. En la «hoja C» se describen operaciones, tanto de tipo lógico como aritmético. Se detallan, por ejemplo, los nombres de los dos operandos que intervienen en una operación normal, el campo resultado...

La cuarta hoja corresponde a la última parte del proceso y es la que nos permite especificar cómo queremos que aparezcan los resultados en la impresora. Es la «hoja O», en la que figuran las descripciones de las líneas de cabecera, las de detalle y las que reflejen totales. También se indican las acciones a efectuar cuando el proceso está en la primera página de la impresión, etc.

En resumen, podemos asimilar el proceso de lectura de la información a las hojas «F» e «I», los procesos a efectuar a la «hoja C» y la presentación de resultados por escrito a la «hoja O». Ese camino de la información es lo que se llama «ciclo lógico de ejecución».

Ficheros del RPGII

En el ciclo lógico del RPGII tiene especial importancia la relación que existe entre los ficheros a tratar. Como en otros lenguajes, existen varios conceptos básicos en relación a los archivos. El CAMPO es la mínima unidad de información. La agrupación de varios campos que tienen una relación se llama *registro*. Y, por último, una cierta cantidad de registros forman un *fichero*.

Para una aplicación determinada, el fichero que contiene la relación principal de registros se llama «Master File». Generalmente reside en dispositivos de alta velocidad de acceso, discos o cintas magnéticas.
Según el método de actualización de información en este fichero maestro, se distinguen los procesos en línea y en modo «BATCH» o «por lotes». Los trabajos que no son críticos para la organización se realizan en «BATCH» y generalmente usando cintas.
El fichero de movimientos suele venir ordenado en relación con el campo clave del fichero *maestro*. El lenguaje RPGII permite que este tipo de relaciones estén muy unidas al ciclo de programación. Cuando se «cambia» de clave, se escribe una línea o un total y se vuelve a comenzar hasta que aparezca una nueva diferencia.
El fichero *maestro* y el de *movimientos* pueden residir ambos en dispositivos de acceso secuencial; por tanto, es necesario que estén ordenados por su clave. Si el *maestro* reside en disco, el acceso ya no tiene por qué ser de este modo. Se leen movimientos del fichero (secundario) y se «encadenan», a través del campo clave, con el fichero *maestro* que es el primario. Y de nuevo se continúa el ciclo.
Si el proceso es en tiempo real, o en línea, la información o los movimientos provienen de las pantallas que están directamente conectadas a la CPU y modifican directamente los registros de este fichero. Para ello es necesario que el acceso sea directo al registro que se quiere modificar. Para ello se usan estructuras de tipo DIRECTO, ISAM o VSAM. Por motivos de seguridad se suelen guardar ficheros con todos los movimientos habidos.

Para saber más

¿Qué significa RPGII?

El lenguaje RPG se desarrolló en los años sesenta por IBM. El interés por parte de los usuarios hizo que se creara en 1974 una nueva versión. Esta mejoraba la potencia del lenguaje y le abría campos más amplios, también trataba de una forma más sencilla las operaciones de entrada/salida, etc. En 1980 apareció el RPGII, diseñado para entornos más sofisticados. Permite usar las técnicas de programación estructuradas y las bases de datos.

¿Por qué no se hacen todas las aplicaciones en tiempo real?

Es función de las prioridades de los trabajos. Si todos requirieran la CPU en todo momento, se retrasaría la finalización de todas las tareas. Por eso se reservan las mayores prioridades a los procesos críticos. Aun así, existen lapsus en los tiempos de respuesta cuando el sistema se sobrecarga. Por otra parte, los procesos BATCH tienen la ventaja de utilizar menos recursos externos o aquellos que no son muy importantes.

¿Cuál es el contenido de las hojas de codificación?

En conjunto, una hoja de codificación consiste en una serie de cuadrículas a rellenar por el usuario. Cada una de ellas tiene impreso su significado. Poseen 80 columnas como las pantallas, o su predecesora la tarjeta perforada. Las dos primeras columnas tienen el número de hoja y de la columna 3 a la 5 el número de línea. La 6 es el tipo de hoja, a partir de la 7 hasta la 75 es distinto para cada hoja. Las cinco últimas contienen el nombre del programa.

Del Pascal al ADA

Los modernos lenguajes evolucionados



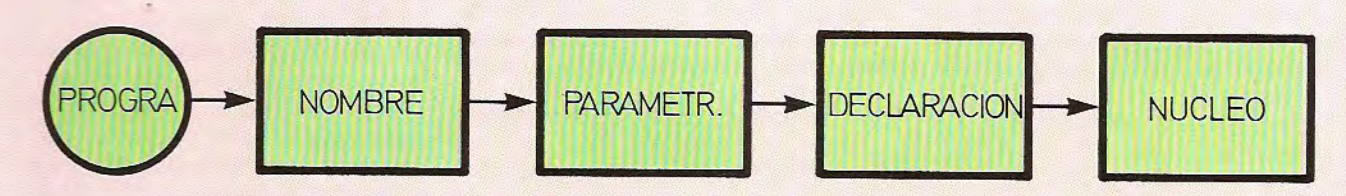
Los lenguajes de alto nivel son cada día más utilizados para el desarrollo de los programas, hasta el punto de haber desplazado a los de ensamble de múltiples cometidos tradicionalmente reservados a lenguajes de programación más próximos a la máquina.

La responsabilidad de este protagonismo recae no sólo en la gran difusión de los tradicionales BASIC, FORTRAN o COBOL, sino primordialmente en la nueva generación de lenguajes de alto nivel nacidos en los años setenta. Ahí está, por ejemplo, el PASCAL —prototipo de los modernos lenguajes estructurados—, el «C» o el ADA, dotados todos ellos de potentes compiladores que permiten obtener una codificación depurada y eficiente.

Habitualmente se citan dos razones en apoyo de los modernos lenguajes de alto nivel frente a los lenguajes más próximos a la realidad de la máquina:

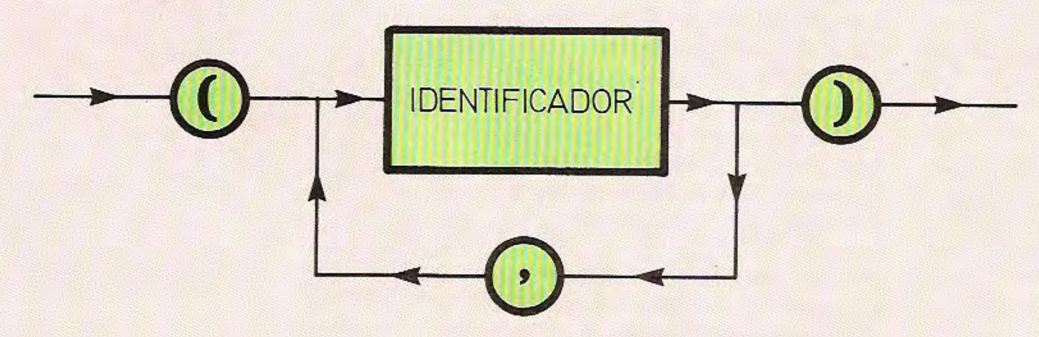
- Parece demostrado que un programador escribe el mismo número de líneas por unidad de tiempo cualquiera que sea el lenguaje empleado. Como un lenguaje de alto nivel incorpora comandos y funciones más potentes que los ensambladores, para desarrollar una misma aplicación se requiere un número menor de programadores.
- El software escrito en un lenguaje de alto nivel permanece muy alejado del





Bloques que componen un programa Pascal. Este debe empezar por la declaración del nombre dado al proceso. Seguidamente se definen los parámetros que van a intervenir. En tercer lugar, se especifican las funciones y procedimientos y, finalmente, se escribe el núcleo.

PARAMETROS



La relativa complejidad de la sintaxis Pascal es el mayor problema con el que se enfrenta cualquier principiante. Los diagramas de Conway constituyen, en este sentido, una gran ayuda. La figura representa un ejemplo de estos diagramas para la definición de parámetros.

hardware. Las sucesivas modificaciones de la máquina implican modificaciones sencillas en los programas y aplicaciones desarrolladas con lenguajes de alto nivel. Estas tienen, por tanto, una vida más larga, y a un precio lógicamente menor.

El lenguaje PASCAL

El lenguaje de programación PASCAL se desarrolló a partir del lenguaje ALGOL. El ALGOL (ALGOrithmic Language: Lenguaje algorítmico) tiene diversas versiones: ALGOL 60, ALGOL 68 (llamados así por haber sido creados en esos años) y ALGOL W. Niklaus Wirst diseñó, en base a ésta última, el PASCAL, lenguaje del que ya existía un compilador en 1970. El objetivo de Wirst era crear un lenguaje muy pedagógico, convirtiendo a la programación en una metodología muy sistematizada. Actualmente, sin embargo, el PASCAL forma parte de las herramientas de trabajo informático disponibles en cualquier mercado de software.



Descripción genérica de un programa Pascal. Cualquier programa escrito en este lenguaje consta, obligatoriamente, de los bloques 1 y 3.

Codificación de los programas PASCAL

La codificación de los programas realizados en PASCAL guarda poca similitud con la codificación realizada en BASIC. Por otra parte, el BASIC puede ser ejecutado tan pronto como es escrito si se emplea un intérprete: éste busca en la línea en curso la instrucción a ejecutar y la traduce saltando a la parte del programa donde se realiza la acción correspondiente. Un programa en PASCAL, sin embargo, no se interpreta: siempre se compila.

Una vez efectuada la compilación, y si no se ha detectado ningún error, el programa está listo para ser ejecutado.

Partes de un programa PASCAL

El lenguaje PASCAL es relativamente moderno. Es por ello que refleja las últimas tendencias en técnicas de programación: presta gran atención a la estructura de los datos, y ofrece ayudas para la definición abstracta de éstos y de la estructura interna del programa.

Un programa PASCAL está dividido en tres partes funcionalmente distintas. La primera parte es la cabecera, en que aparece el nombre del programa, precedido de la palabra PROGRAM, y los nombres de los ficheros o dispositivos implicados.

La segunda parte se compone de la descripción de los datos: en esta zona se definen todas las variables, constantes y etiquetas utilizadas por el programa.

La tercera zona comprende el conjunto de sentencias a ejecutar, y que utilizan los datos y procedimientos creados anteriormente. Esta sección se emplaza entre las palabras reservadas BEGIN y END. Cuando se hace referencia a datos no definidos anteriormente, la compilación señala un error.

Variables del programa

La descripción de los datos de un programa PASCAL depende de la estructura o función que vayan a tener durante el proceso, o del criterio del programador:

LABEL indica el número entero correspondiente a la línea donde se coloca una instrucción GOTO.

CONST asigna a un identificador un valor constante.

TYPE define el tipo de valor que puede tener una variable dada. Esta puede pertenecer a cualquiera de estas clases:

- Variables simples estándar:

— INTEGER: números enteros.

— CHAR: caracteres codificados en ASCII.

— BOOLEAN: las variables pueden tomar los valores lógicos «True» y «False».

— REAL: números reales.

- Variables simples escalares: El conjunto de valores que puede tomar la variable así definida coincide con un conjunto de datos enumerados por el programador. Por ejemplo, TYPE COLOR = (blanco, azul, negro, rojo) asigna a la variable definida como COLOR el conjunto de estos cuatro colores.

- Clase estructurada: Las estructuras de tipo simple se combinan mediante los comandos UNPACKED y PACKED.

— ARRAY define una matriz de valores por un tipo base y mediante un índice escalar.

Ejemplo: TYPE mes = ARRAY (1 ... 31) of INTEGER.

— RECORD define a una estructura con un número fijo de «campos», cada uno de los cuales puede ser de un tipo distinto. Esto último es una diferencia con respecto al Array, en donde cada uno de los valores es del mismo tipo. Por otra parte, el acceso a cada uno de los campos de un Record se efectúa mediante el nombre del propio registro y de su campo.

Un ejemplo puede ser:

TYPE familia = RECORD
Miembro: (padre, madre, hijo, hija)
Cantidad: INTEGER
Media: REAL
END

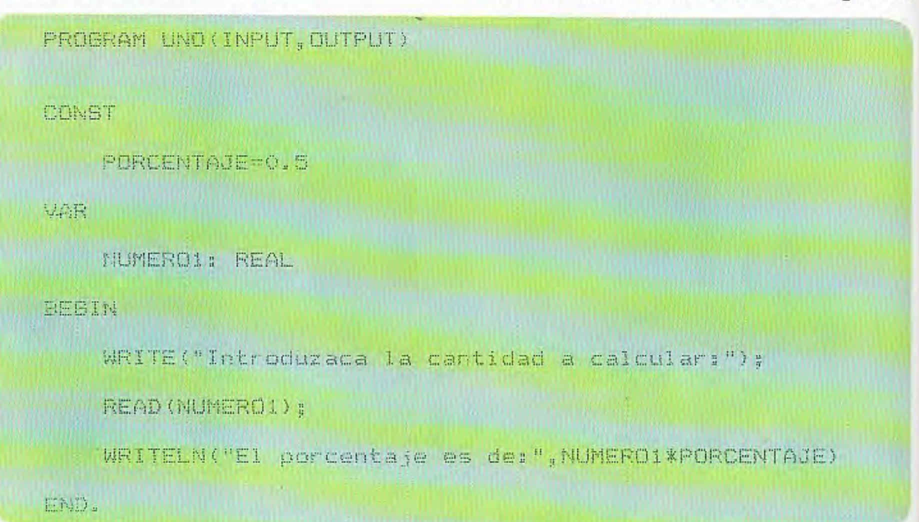
Es posible, además, establecer una estructura más flexible de cada registro variando el tipo y la cantidad de los campos mediante la instrucción CASE ... OF).

— SET sirve para crear y definir un conjunto de elementos del mismo tipo. Como tal conjunto permite la realización de operaciones propias en este tipo de estructura matemática, como la intersección, suma, etc.

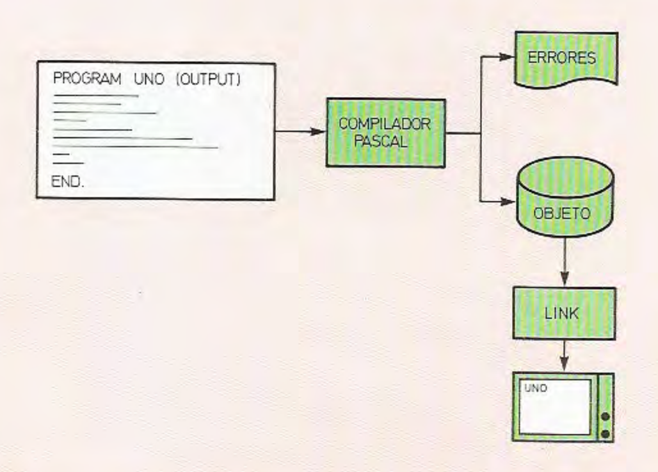
— FILE organiza los ficheros en forma secuencial. Estos se componen de elementos del mismo tipo.

La instrucción de posicionamiento se coloca al principio del fichero en la forma RESET (fichero). La de lectura y posicionamiento en el siguiente registro tiene el formato READ (fichero, a). La condición de final de fichero se compone mediante la función EOF (fichero).

— POINTER (↑) se utiliza en estructuras de tipo dinámico para la gestión de variables reubicables durante la ejecución del programa. Este comando permite almacenar la información en forma no secuencial. El acceso a los registros



Ejemplo de programa Pascal. En el encabezamiento se especifica el nombre del programa «UNO» y el nombre de los ficheros usados, «INPUT» Y «OUTPUT».



Esquema del proceso seguido por un compilador para obtener el código objeto de un programa fuente Pascal.

de un fichero se obtiene en forma ordenada.

Para definir las variables se utiliza la palabra reservada VAR y, a continuación, se especifican, separadas por comas, las variables y el entorno al que pertenecen.

El lenguaje LOGO

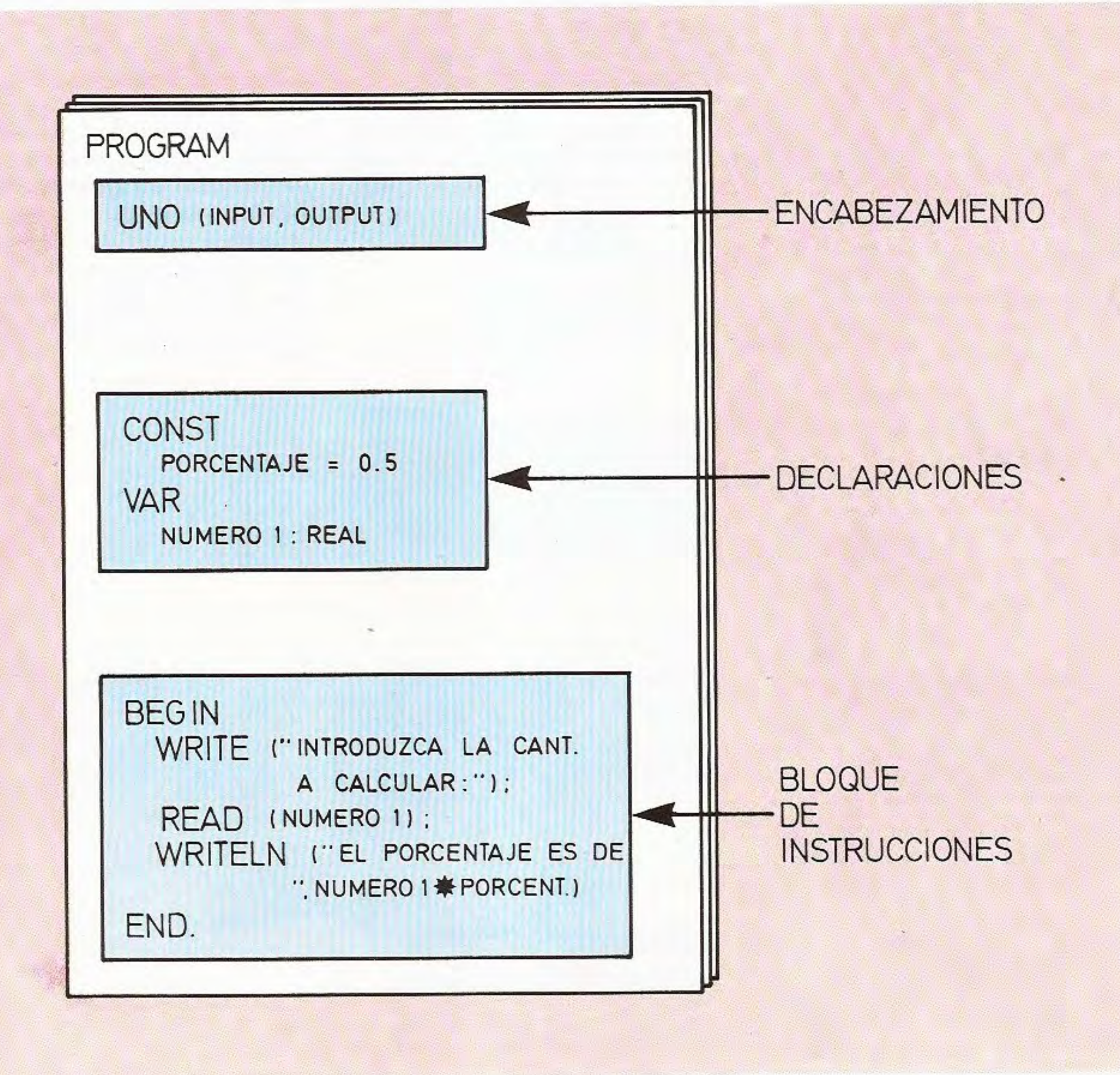
El LOGO es un lenguaje que tiene algo más de una década. Fue desarrollado por Seymour Papert del MIT. La finalidad principal era la de acercar a los niños el mundo de los ordenadores.

Lo que más llama la atención de este lenguaje es la existencia de la llamada «tortuga», que se mueve por la pantalla del ordenador. Un concepto un tanto extraño para los que conocen otros lenguajes, pero que resulta muy «amigable» para un niño.

El origen de esta tortuga está en un artilugio de tipo mecánico que poseía ruedas y que era controlado a través de un teclado. Fue diseñada por W. Grey Walter al final de los cuarenta.

El LOGO es un lenguaje informático normal, y no por ser para la enseñanza o para un entorno de niños significa que se trata de un lenguaje algo secundario y sin utilidad real.

Posee la forma propia de la metodología de procedimientos (bloques de programas ejecutables independientemente o no), e introduce la idea de creación de instrucciones «propias y personales», en base a las primitivas del LOGO.



Bloques constituyentes del programa ejemplo «UNO». Al encabezamiento le sigue la declaración de variables y constantes, tras lo cual se escribe el núcleo, o bloque de instrucciones.

```
1   CLEAR SCREEN
2   ENTER A
3   ENTER B
4   HIRES
5   MOVE TO 100-160
6   PEN DOWN
7   LABEL FIGURA2
8   FORWARD A
9   ROTATE RIGHT B
10   CALCULATE A=A+3
11   USE FIGURA2
12   ROUTINE END
```

Esto es un ejemplo de programa que utiliza la recursividad. El procedimiento, figura 2, se usa a sí mismo. El parámetro que cambia es el contenido de la variable A. Puede observarse que no existe forma para salir del ciclo (valores aconsejados 20, 135; 20, 90).

Es interactivo, cosa fundamental para la enseñanza.

Soporta la recursividad, es decir, normalmente un procedimiento puede incluir la llamada a otro, pero si lo que hace es llamarse a sí mismo, aparece la llamada recursividad. Existe lo que se llaman palabras y listas. Una palabra es cualquier sucesión de letras, números u otros caracteres, excepto el blanco. Una lista es un conjunto de palabras o de listas, el tamaño de cada elemento puede ser variable. Como no es necesario definir los tipos de variables, durante un mismo programa puede usarse una cierta palabra primero como número, después como variable alfanumérica y luego como número otra vez.

Pero, quizá, lo que más atraiga y estimule y, por tanto, aporte un mayor interés en el aprendizaje del niño, sea la parte gráfica y sonora del lenguaje.

El lenguaje «C»

A finales de los años sesenta, en el MIT (Massachusetts Institute of Techonology) se estaba trabajando en un proyecto para mejorar las técnicas de software en un sistema de tiempo compartido.

Colaboraban con el MIT las firmas BELL, GE y HONEYWELL. El proyecto se tradujo en el sistema operativo MULTICS, pero la firma BELL se retiró del proyecto al no ser el resultado concordante con lo esperado por ella.

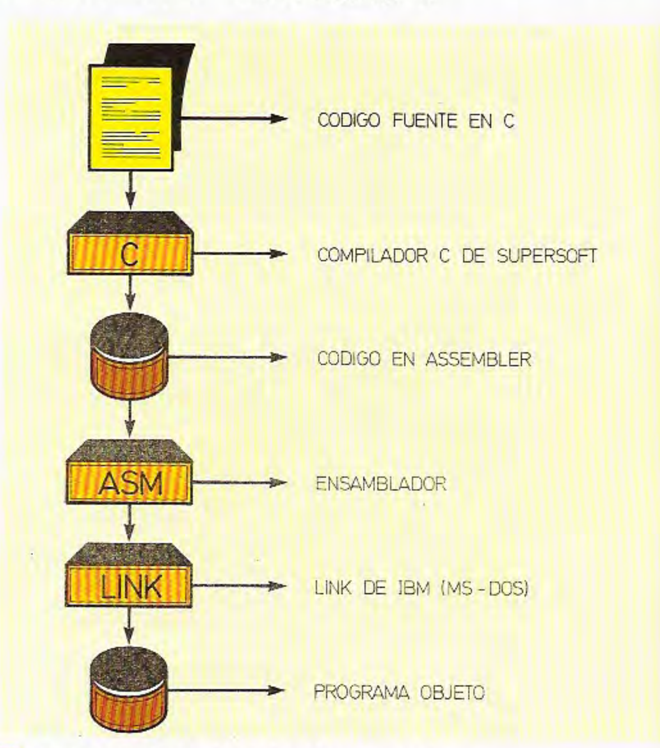
Uno de sus científicos, Keen Thompson, que tenía algún software hecho para el MULTICS, se dedicó a escribir un nuevo programa de base, haciéndolo en el lenguaje ensamblador del ordenador PDP-7.

Este sería el punto de partida del sistema operativo UNIX. Otro científico, Dennis Ritchie, se dedicó, también, a desarrollar el UNIX y poco a poco surgieron correcciones y aumentos de este sistema operativo.

Enterado posteriormente Dennis Ritchie de la existencia de un intérprete llamado B, dedicó sus esfuerzos a mejorarlo y aumentarlo, surgiendo así el llamado lenguaje C.



El lenguaje C, que en principio fue diseñado para grandes ordenadores, se ha revelado como una herramienta excepcional para su utilización en los modernos microordenadores.



En la figura se describe el proceso de compilación de un programa en lenguaje C bajo el compilador SuperSoft utilizado en el ordenador personal de IBM.

El paso siguiente fue entonces reescribir el sistema operativo UNIX en C, ya que éste resultó ser un lenguaje ideal para la programación de sistemas. El C es un lenguaje de propósito general, es decir, no está orientado específicamente a la resolución de un cierto tipo de problemas, del que puede decirse que no es un lenguaje de alto nivel tipo BASIC, PASCAL, RPGII, etc., ni, por supuesto, tampoco de bajo nivel como el ensamblador. Pero, sin embargo, reúne las ventajas de ambos tipos.

Es de destacar que C no está sometido a ningún sistema operativo determinado y se caracteriza por permitir una gran transportabilidad (utilización en distintos ordenadores) de los programas que se escriben en este lenguaje. Conviene señalar aquí, no obstante, que cada implementación tiene su propia biblioteca de funciones de entrada-salida, que sí son propias de cada máquina.

Al ser el lenguaje C, por decirlo de algún modo, de un cierto bajo nivel, no tiene alguna de las características que en lenguajes como BASIC o PASCAL son naturales. Así, no existe el tratamiento de *strings* o de matrices tal como se entienden en estos otros lenguajes, y no aparecen métodos sofisticados de accesos ni de controles de flujo.

Estas tareas se realizan mediante funciones definidas para estos casos. La otra cara de la moneda es que sí soporta las técnicas más normales de la programación estructurada.

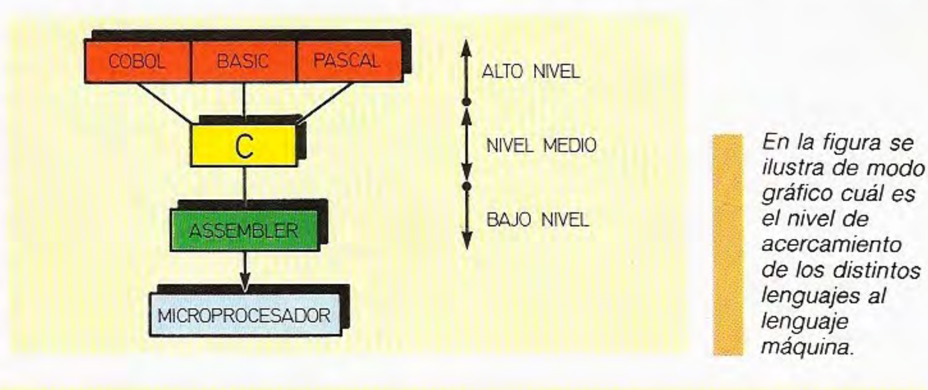
Concepto de OBJETO

Al actuar el C como un lenguaje de bajo nivel, trata de la misma forma a los diferentes tipos de variables, como son letras, dígitos, direcciones de cualquier tipo, etc. Aparece entonces el concepto de OBJETO: área de memoria que contiene un tipo de dato determinado.

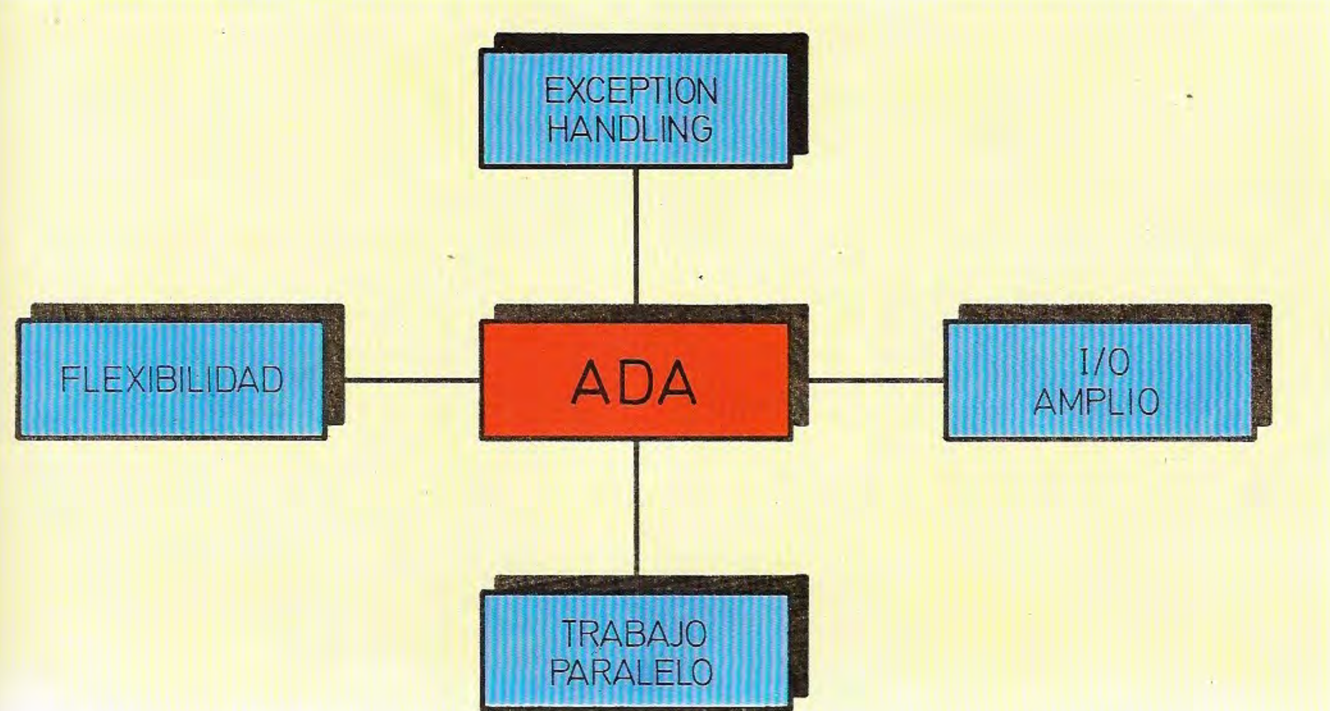
Existen cuatro tipos fundamentales de objetos:

— CHART, IN, FLOAT y DOUBLE.

- *CHAR:* Un objeto CHAR es un espacio de almacenamiento que ubica un elemento del set de caracteres del que dispone la máquina.
- *INT:* Un objeto INT es un área de



En la figura se ilustra de modo gráfico cuál es el nivel de acercamiento de los distintos lenguajes al lenguaje máquina.



Las características fundamentales del lenguaje ADA son: gran flexibilidad, control de cualquier posible error durante el tiempo de ejecución, posibilidad de proceso en paralelo y gran capacidad para entradas y salidas de periféricos.

memoria que almacena un número de tipo entero (según el tipo de procesador, éste será de 8, 16 o más bits). Este tipo puede ser de varias clases: SHORT, LONG y UNSIGNED.

- *FLOAT:* Un objeto FLOAT es un área de almacenamiento que guarda un número en coma flotante y que es de simple precisión.

- *DOUBLE:* Un objeto DOUBLE es un área de memoria que contiene un número de doble precisión en coma flotante.

Estos tipos de objetos se pueden combinar con los siguientes tipos de datos para formar estructuras:

— Matrices.
— Funciones que devuelven un objeto de cualquier tipo.
— Punteros a los varios tipos de objetos.
— Datos que relacionan a dos o más tipos de objetos.

Definidas ya las posiciones de los datos, el paso siguiente es dar nombre a estas posiciones para poder tratarlas de forma simbólica. Siguiendo las convenciones normales: las letras serán de la «a» a la «z», los números del 0 al 9, y se usará el subrayado. Dependiendo del compilador, el máximo número de caracteres significativos será ocho, seis, etc.

Clases de almacenamiento

Existen cuatro clases de almacenamiento:

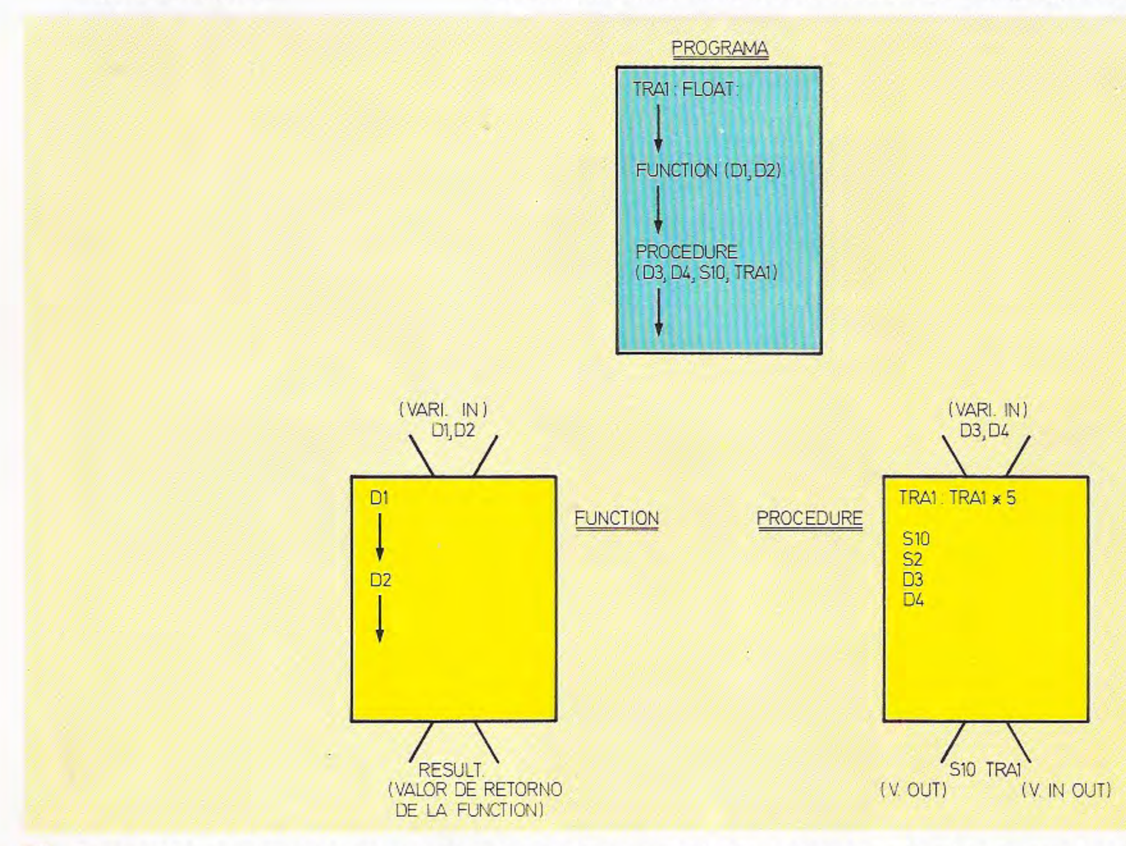
- *AUTO* (automática): Se conocen con esta denominación los objetos que sólo están asociados a una función o bloque de programa y que se pierden al salir de ellos.

Es decir, dentro de una función, por ejemplo, la variable PRUE tiene un significado, mientras que si el flujo pasa por otra función la variable PRUE comienza con un valor que es independiente del anterior que tuviera. Esta es la opción que se escoge para todas aquellas variables que no tengan otra especificación.

- *STATIC:* Es muy parecido a AUTO, diferenciándose en que al salir de una función o bloque determinado, la variable sigue con el valor que tenía.

- *EXTER:* Son las variables que están disponibles a lo largo de todo el programa, aun entrando en funciones o bloques.

- *REGISTER:* Una variable de este tipo significa que se almacena en alguno de los registros internos del procesador, para así hacer más rápida la ejecución del programa. Las variables de este tipo



En el gráfico puede observarse, dentro de un programa, cuál es la diferencia entre función y procedimiento. Como puede verse, en un procedimiento se puede tener: la variable TRA 1, que es de tipo entrada/salida, las variables D3 y D4 que son de entrada y la variable S10 que es de salida. Por el contrario, en una función se dispone de unas variables de entrada (D1 y D2) y se obtiene un resultado que es el valor de retorno de la función.

no pueden exceder la capacidad propia de los registros del procesador y, además, cumplen las especificaciones del almacenamiento AUTO.

Programas en lenguaje C

Por lo que respecta a las instrucciones del repertorio del lenguaje C, hay que señalar que su número es bastante limitado. La estructura de programación más potente es la de función, cuyas partes más importantes son: el nombre, los parámetros de entrada, los de salida y los de retorno.

Las sentencias siguen la forma normal utilizada en otros lenguajes, aunque se pueden crear grupos de sentencias, que actúan como una sola, simplemente encerrándolas entre llaves.

Un programa en lenguaje C está formado por un conjunto de funciones entre las cuales una sola lleva el nombre «main»: es la función principal o maestra.

Ejemplo:
```
Main ()
{
Float número;
número = 3*2,3 + 5
}
```

Este es un programa escrito en lenguaje C formado por una sola función: «main ()». Esta función da valor a una variable «número» que es de tipo «float» (es decir, real con coma flotante y de simple precisión). Conviene señalar aquí que en todo programa escrito en lenguaje C se deben cumplir las siguientes características:

- A continuación del nombre de toda función siempre se colocarán paréntesis, entre los que se escriben, si existen, los argumentos.
- El cuerpo de toda función empieza y termina con una llave.
- Toda función debe ser definida previamente.
- Las funciones terminan, generalmente, con punto y coma (;).

PROCEDURE IS

BEGIN

END

NOMBRE DE LA PROCEDURE (PROG.)

PARTE DE DECLARACION QUE CONTIENE LA DESCRIPCION DE LOS DATOS

PARTE DE INSTRUCCIONES, DESCRIPCION DE LAS OPERACIONES

La estructura básica de todo programa escrito en lenguaje ADA, adquiere la configuración que muestra el diagrama de la figura.

El lenguaje ADA

El nombre ADA no hace referencia a unas oscuras siglas ni tampoco es un nombre tomado al azar.

Es, sencillamente, el nombre de la que se considera, históricamente, la primera mujer programadora, Augusta Ada Lovelace (hija de lord Byron).

Ella fue la que, tras estudiar las posibilidades de la máquina de Babbage, creó la idea de programa y las estructuras de subprogramas y módulos para evitar, como sucedía en el diseño original, la repetición de instrucciones.

El lenguaje ADA nació cuando el Departamento de Defensa USA se propuso implantar un único lenguaje que pudiera sustituir a sus casi 400 lenguajes de programación. Hay que tener en cuenta la gran disparidad de «objetos programables» que pueden equipar a un ejército moderno.

El diseño se especificó de forma que se cumplieran una serie de requisitos que proporcionaba una comisión especial de este Departamento. Se presentaron a concurso varios grupos cuyas soluciones se desestimaron. La que pasó la selección final fue la presentada por Cii Honeywell Bull (Francia) y que había sido diseñada básicamente por Jean Ichbiah en el año 1978.

Una segunda versión del lenguaje ADA fue presentada en 1980. En 1983 dicho lenguaje fue considerado estándar por ANSI y es también un estándar militar, hasta el punto de que ADA es

una marca registrada por el Gobierno USA.

Señalemos que un compilador debe pasar unos 2.000 tests para llegar a ser aceptado por una oficina y dar un certificado de la validez del compilador.

Actualmente existen pocos compiladores reales, aunque hay muchos compiladores parciales.

Los requerimientos del Departamento de Defensa exigían que fuera un lenguaje muy legible, que permitiera con mucha facilidad el trabajo con subprogramas y la programación modular, que tuviera facilidades para entradas/salidas por muchos dispositivos (hay que considerar que debe poder controlar multitud de dispositivos militares, misiles, sistemas básicos de control, etc.), adecuado para el proceso en paralelo y facultado para controlar cualquier posible tipo de error durante el tiempo de ejecución.

Esta última es una característica muy importante en el manejo de dispositivos de precisión.

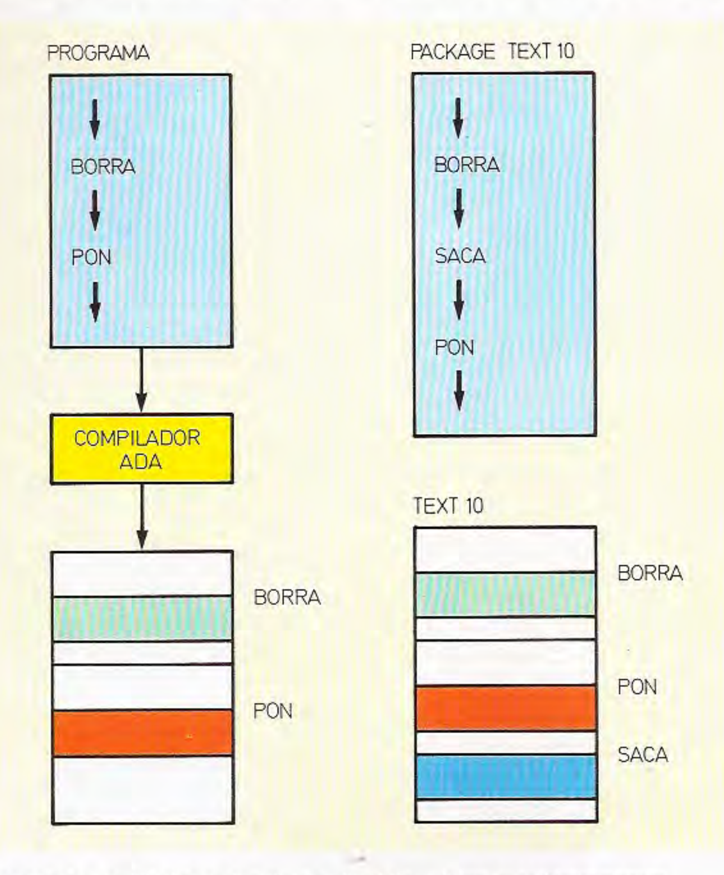
Estructura de un programa en lenguaje ADA

Muchas de las características del ADA son similares a las de otros lenguajes de alto nivel, como el PASCAL y otros menos conocidos.

Un programa escrito en ADA puede adoptar, en general, una estructura como la que se indica en el cuadro adjunto.

Todo programa o subprograma contenido dentro de un programa principal seguirá esta configuración.

Obsérvese que los términos *procedure, is, beguin* y *end* conforman la estructura básica de la descripción del programa.



El concepto de Package (conjunto de datos, tipos y subprogramas que pueden ser utilizados en otros programas), resulta de gran utilidad en el momento de la compilación.



Errores en ADA

Durante el curso de la ejecución de un programa pueden producirse muchos errores, los cuales puede que no sean achacables al propio programa, sino que se deban a la entrada de datos. Por ejemplo, que los datos estén fuera de rango. Al producirse un error, la simple detención del programa no es una solución aceptable en el marco de aplicación del ADA. Al efecto se han incluido en el referido lenguaje procedimientos para intèntar su resolución; surge así el concepto de EXCEPTION. En tiempo de ejecución sólo se admiten cinco tipos de EXCEPTION:

— NUMERIC_ ERROR: cuando en el momento de realizar una operación de tipo aritmético el resultado está fuera de rango o no coincide con la precisión de la variable para recibir el resultado.

— CONSTRAIT_ ERROR: usada para los errores de fuera de rango de los índices o elementos nulos.

— PROGRAM_ ERROR: Se da para los errores de lógica en el programa; por ejemplo, en un select no se cubrieron todos los casos posibles y no existe un else que recoja todas las restantes situaciones. También surge cuando aparece una referencia a zonas del programa no definidas.

— STORAGE_ ERROR: se produce en relación, básicamente, con el espacio de almacenamiento; por ejemplo, cuando se excede la capacidad existente en el momento de la creación y ubicación en memoria.

— TASKING_ ERROR: sucede cuando entre tareas existen problemas de comunicación.

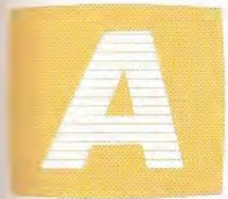
Trabajos en paralelo

El concepto de trabajo en paralelo significa que una tarea (task) puede ser ejecutada concurrentemente con otra. Esta unidad de programa se parece mucho a un PROCEDURE, diferenciándose en que el cuerpo de una tarea no se ejecuta si no es activado por otro programa.

Compilación separada

El lenguaje ADA permite la posibilidad de la compilación separada. Los subprogramas y los PACKAGES pueden ser compilados separadamente. Como las unidades de programa se construyen en base a una descripción del interface y un cuerpo, es posible compilar separadamente los cuerpos, tanto de subprogramas como de PACKAGES.

Traducción de los lenguajes



Al alejarnos de la máquina programando con lenguajes de alto nivel, se hace patente la necesidad de contar con programas especializados que faciliten y apoyen la tarea del programador. Estos programas forman parte del software del sistema y suelen ser suministrados por el propio fabricante del ordenador.

Los compiladores

El incremento del uso de las macroinstrucciones y su constante sofisticación hizo que en los programas confeccionados para distintos equipos se encontraran muchas funciones comunes, tales como leer datos de un fichero en disco o escribir en una cinta magnética. El análisis de estas funciones comunes llevó al desarrollo de los lenguajes de alto nivel.

Para que los programas escritos en

Ayudas al proceso de programas

estos lenguajes puedan ser ejecutados por el ordenador, es preciso convertirlos previamente en programas objeto, codificados en lenguaje máquina. Este proceso de conversión recibe el nombre de compilado o *compilación* del programa fuente.

El *compilador* es el programa auxiliar que controla el proceso de *compilación*, realizando las siguientes funciones:

- Leer las instrucciones del programa fuente, a través de un periférico de entrada.

Para que la programación en lenguaje de alto nivel sea eficaz es preciso contar con determinados programas auxiliares que trasladen la información de origen a un lenguaje inteligible por el ordenador.

- Clasificarlas por número de secuencia de las instrucciones.
- Convertir las macro y microinstrucciones a instrucciones en código de máquina.
- Crear la tabla de direcciones de memoria de las referencias (variables, subrutinas, áreas de datos).
- Producir el programa objeto en un soporte de almacenamiento masivo.
- Editar un listado, tanto del programa fuente como del objeto.
- Detectar los errores sintácticos del programa.

El proceso de compilación se repite hasta que se obtenga la llamada *compilación limpia,* es decir, una compilación exenta de errores.

Cada lenguaje de alto nivel necesita un compilador para cada tipo de ordenador en el que vayan a ser ejecutados los programas, ya que los respectivos lenguajes de máquina son distintos.

Intérpretes

El principal inconveniente asociado al empleo de los compiladores es que para ejecutar el programa es preciso compilarlo previamente; es decir, se trata de un proceso «batch».

Los intérpretes resuelven este problema, ya que traducen, «interpretan» y procesan las instrucciones según se van introduciendo; como resultado de esta ejecución van almacenando datos y vi-

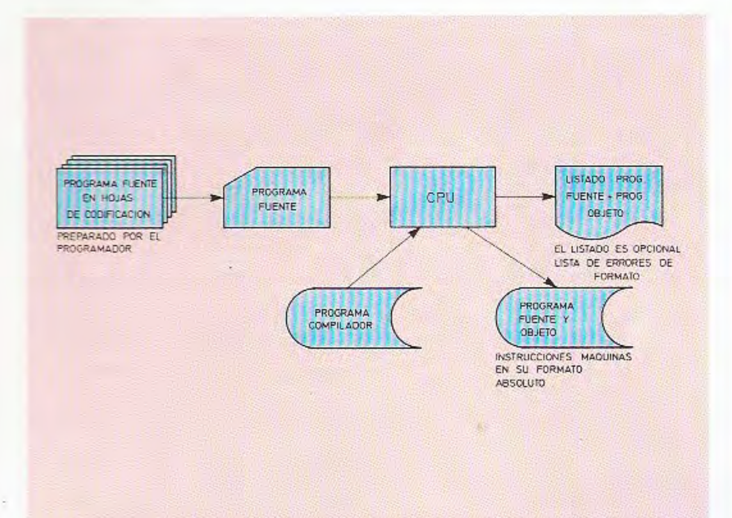
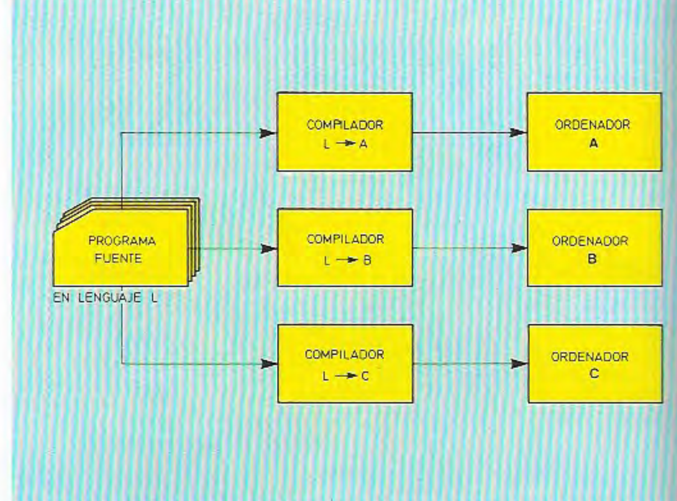
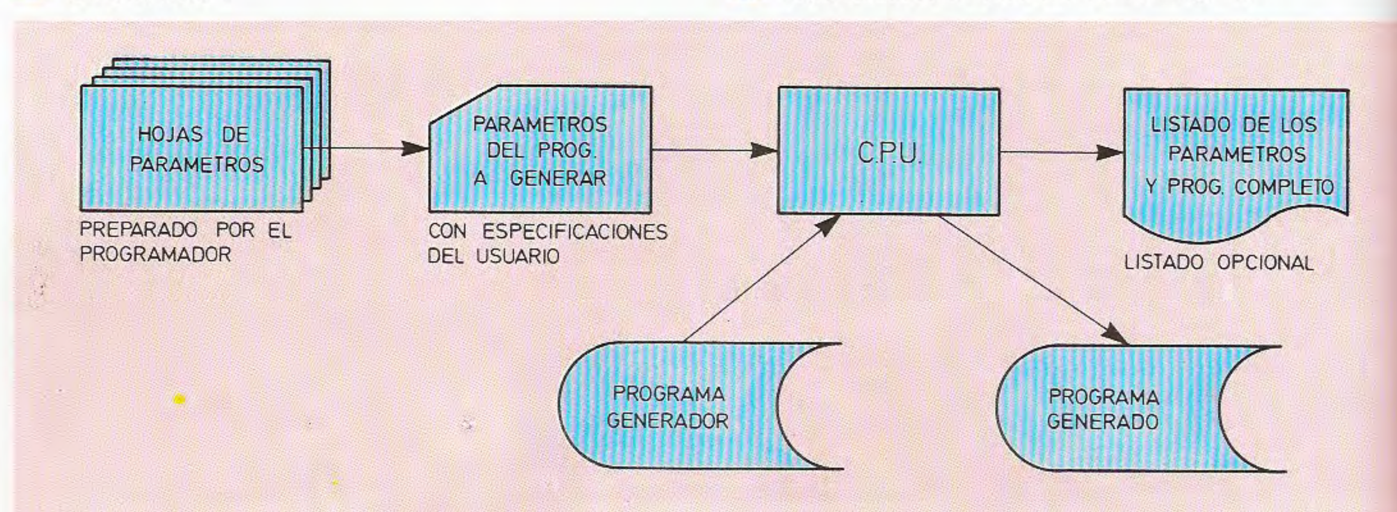


Diagrama representativo de un proceso de compilación



Para que un programa pueda ser procesado en distintos ordenadores es necesario contar con compiladores especializados en cada una de las máquinas.

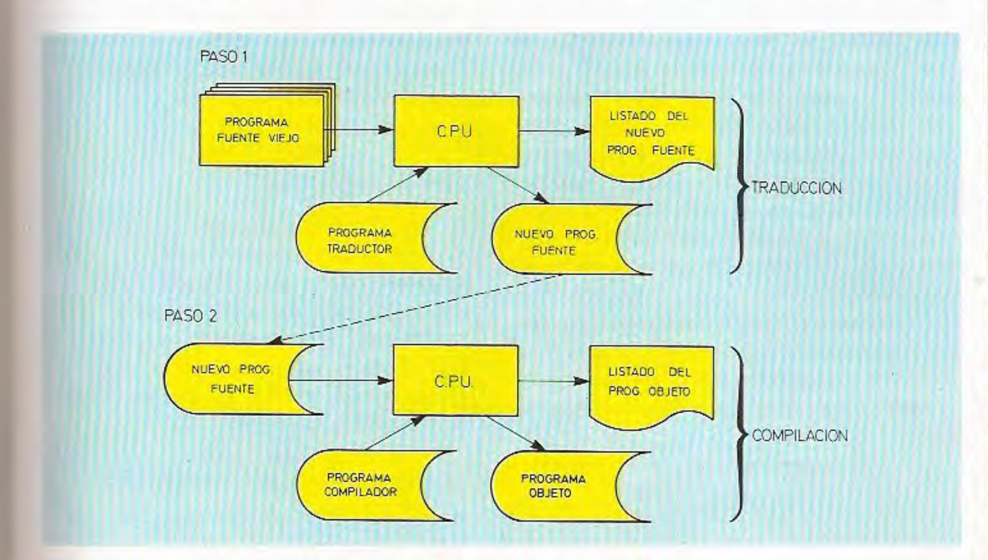


El proceso de generación de un programa es semejante al de «compilación»; no obstante, la información de entrada no será un programa, sino únicamente los parámetros definitorios del programa a generar.

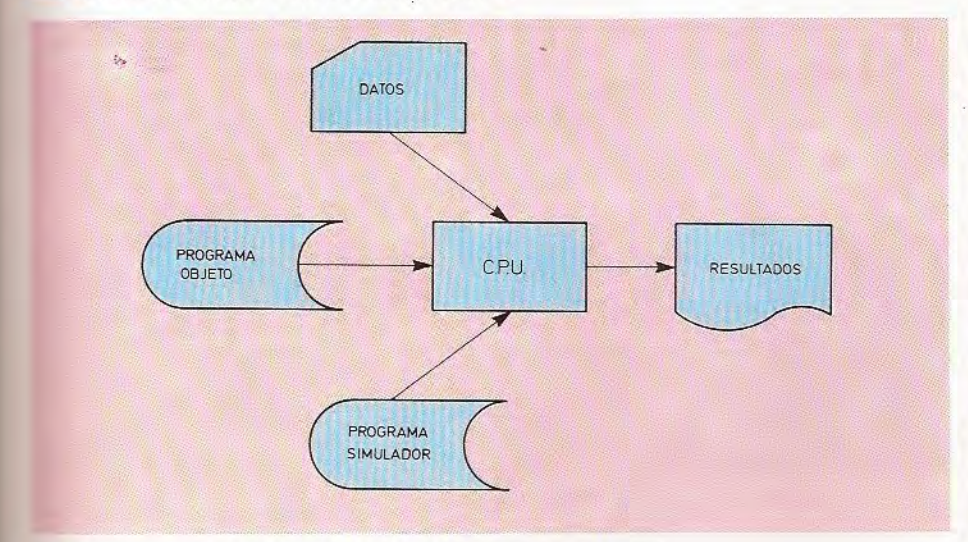
sualizando los resultados progresivamente.

Generadores de programas

Existen procesos cuya lógica se puede repetir con frecuencia dentro de los programas, como son la clasificación de registros, la visualización de informes, el formateado de resultados, etc. Para evitar la necesidad de reprogramar constantemente este tipo de procesos existen los denominados *generadores*. Un *generador* es un programa capaz de construir otros programas, utilizando *parámetros* que dependen de cada caso particular. Por ejemplo, a un programa generador de clasificaciones sólo es necesario darle parámetros, tales como: dónde se encuentran las claves de clasificación dentro del registro y cuál es su



Los programas traductores convierten un programa fuente en otro programa fuente que, posteriormente, debe ser sometido a un proceso de «compilación».



Para que sea posible utilizar los programas en un nuevo ordenador distinto del original, puede recurrirse a los programas auxiliares denominados «simuladores».

La máquina virtual

El concepto de firmware nos sirve como elemento de introducción a la máquina virtual. Recordemos que el firmware se define como un conjunto de microprogramas residentes permanentemente en la máquina y al alcance inmediato de la CPU.
Cambiando el firmware de un ordenador, cambian realmente las características del mismo. El firmware hace posible transformar, por ejemplo, un ordenador de gestión en uno de tipo científico. Para realizar esta transformación, lo único que hay que hacer es cargar en el ordenador de gestión un firmware que posea características propias de un ordenador científico, como puede ser la de operar en coma decimal flotante; sin necesidad de incorporar nuevos elementos hardware se consigue que el ordenador opere con un gran rendimiento, tanto en el aspecto comercial como en el científico.
La introducción del firmware oportuno hace que un ordenador compile y ejecute un programa con mayor rapidez y utilice menos memoria interna. Hasta ahora, al adquirir un nuevo ordenador resultaba obligado modificar los programas fuente para pasarlos al lenguaje del nuevo ordenador. Un firmware apropiado permite que los programas objeto existentes puedan ser ejecutados sin recompilarlos y el nuevo ordenador actuará como si fuera el antiguo; esto es, operará VIRTUALMENTE de la misma manera que el ordenador antiguo. Tenemos, pues, una máquina virtual.
Los programas fuente escritos en un lenguaje como el COBOL pueden compilarse y ejecutarse en una máquina virtual COBOL, que actuará como si se hubiera diseñado para cumplir los requisitos del COBOL.
De lo dicho hasta ahora se saca una conclusión importante para el mundo informático. Una máquina puede transformarse en distintas máquinas virtuales, a medida que el usuario tenga necesidad de ello, cargando diferentes firmwares. Este puede hacer trabajar a su ordenador como una máquina virtual COBOL, o como una máquina virtual FORTRAN, e incluso como una máquina de proceso de comunicaciones conectándole terminales.
Una ventaja muy importante para el usuario de un ordenador con firmware es que puede ampliar o cambiar su ordenador convirtiéndolo en un sistema más potente con un coste y un esfuerzo moderados.

longitud, nombre del archivo, tamaño del bloque, longitud del registro, etc.

Traductores

Otro inconveniente de los programas escritos en lenguaje de alto nivel es el que supone su traslado a otro ordenador, dentro de cuyo software no existe un compilador para el lenguaje en el que están escritos los programas.

Reprogramar todas las aplicaciones puede ser no sólo muy costoso, sino prácticamente inviable.

Los programas *traductores* convierten las instrucciones fuente de un lenguaje en las equivalentes instrucciones fuente de un segundo lenguaje. Este nuevo programa fuente puede ser compilado.

El uso de los traductores reduce el tiempo necesario y el coste de la puesta en marcha de un nuevo ordenador. No se pueden utilizar si durante el mantenimiento de los programas antiguos se han realizado «parches» (patching), a no ser que las modificaciones se hubieran introducido también en el programa fuente.

Simuladores

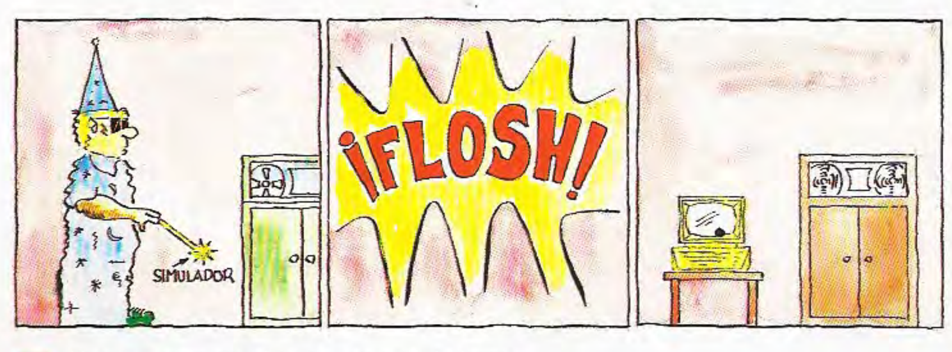
Otra solución para seguir utilizando los mismos programas en un nuevo ordenador es el uso de programas *simuladores*.

Un programa *simulador* logra que un ordenador actúe como si fuera otro distinto. La recepción, tratamiento y salida de datos aparentemente es igual que con el ordenador simulado.

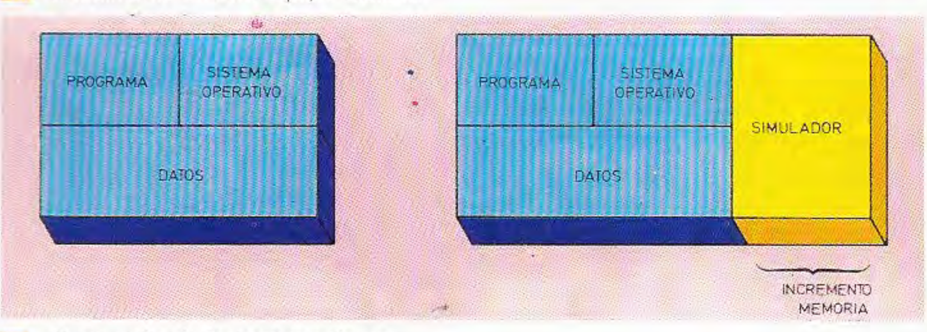
A diferencia del programa traductor que trabaja con el programa fuente, el simulador opera con el programa objeto.

El principal inconveniente de los programas simuladores es que aumentan mucho el tiempo de proceso y necesitan más memoria, ya que el programa simulador debe permanecer en memoria junto con el programa objeto que va a ser procesado.

De todas formas, estos inconvenientes pueden ser resueltos con soluciones «firmware», que pueden convertir los ordenadores en «máquinas virtuales».



Un programa simulador es capaz de lograr que un determinado ordenador se convierta virtualmente en otro equipo distinto.



Ocupación de la memoria de un ordenador sin y con la presencia de un programa simulador.



Para saber más

¿Por qué se llama compilador?

Porque una de las técnicas empleadas en el análisis lexicográfico y sintáctico es la de pilas.

¿Cuál es la diferencia entre un traductor y un compilador?

El traductor (en inglés «translator») convierte un programa fuente escrito en un determinado lenguaje en otro programa fuente en distinto lenguaje. Por su parte, el compilador convierte un programa fuente en un programa objeto en código de máquina.

¿Qué es un «parche»?

Cuando es necesario hacer alguna modificación en un programa compilado se suelen introducir las instrucciones oportunas en el código de máquina correspondiente. Se dice que el programa se ha «parcheado». Hay que tener cuidado puesto que, una vez «parcheado», el programa objeto ya no es la traducción directa del programa fuente original.

¿El RPG es un lenguaje o un generador?

RPG son las iniciales de Report Program Generator. Originalmente se diseñó para proceso de salida de informes impresos. El programador usaba hojas específicas en las que definía la entrada y la salida. Posteriormente se expandió hasta convertirse en un verdadero lenguaje de programación que permite aplicarlo a problemas complejos.

Estructura de los programas

Saltos, bucles y tomas de decisión



El método de funcionamiento habitual de la unidad central de un ordenador consiste en la ejecución secuencial de las instrucciones almacenadas en la memoria, en el orden preciso en que se encuentran. Este método primario no es exclusivo, sino que existen algunas técnicas de programación que conducen a otras estructuras; las tres técnicas que generan las estructuras más conocidas y de uso más frecuente son los *saltos*, los *bucles* y las *decisiones múltiples*.

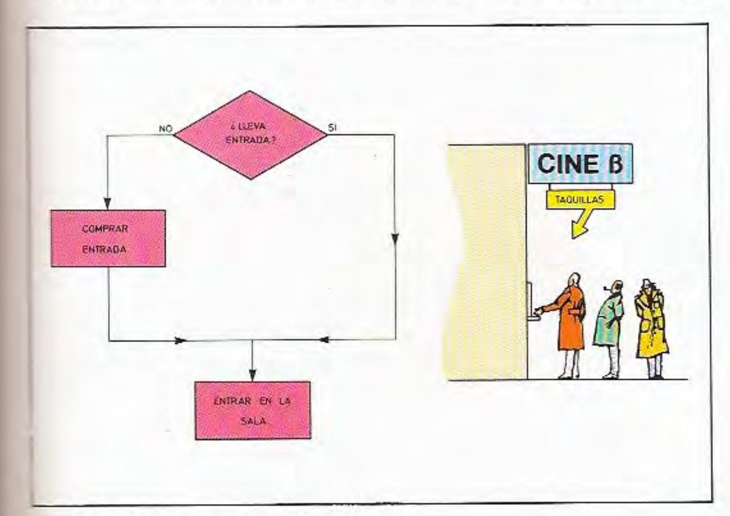
Los saltos

Los saltos forman parte de las técnicas de programación que conducen a estructuras de programas no totalmente secuenciales. Con las instrucciones de SALTO se logra que la ejecución del programa se desarrolle de una u otra forma, de acuerdo con decisiones lógicas tomadas en función de los datos o resultados anteriores. Existen dos tipos de saltos: incondicionales y condicionales.

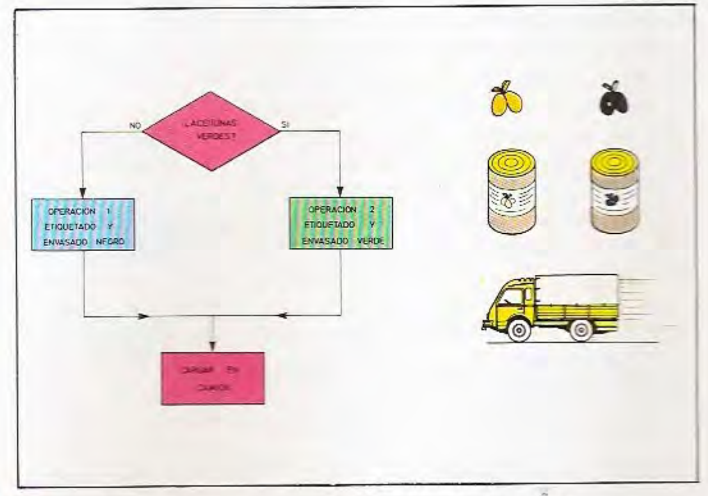
1. *Salto incondicional*

Al llegar a una instrucción de esta índole, el programa rompe obligatoriamente la secuencia normal prosiguiendo la ejecución en otro punto del programa.

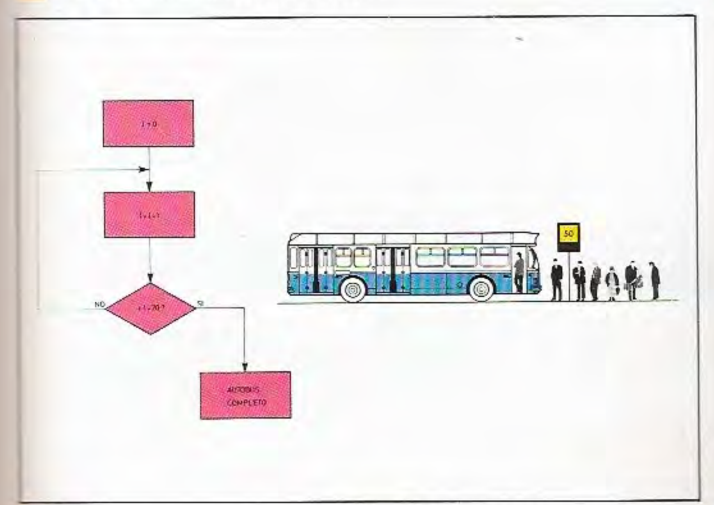
Expliquémoslo con un ejemplo. Si vamos en un coche por la carretera y nos



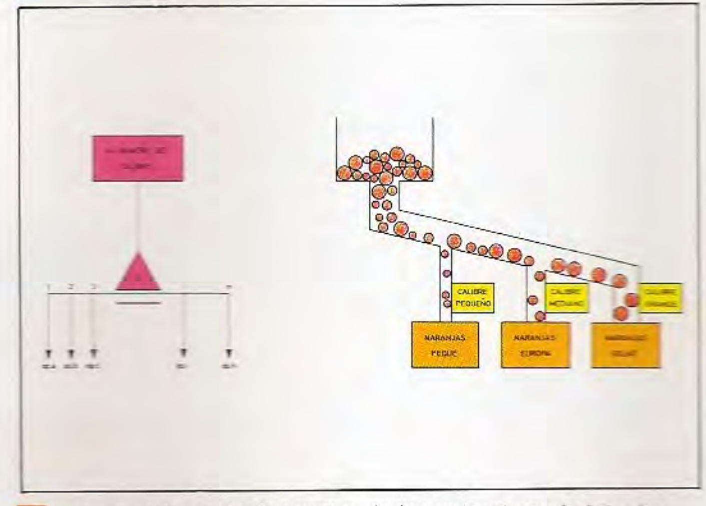
La entrada en una sala de cine sigue un proceso cuya estructura corresponde a la de un «salto condicional».



La condición impuesta para el salto condicional puede derivar el proceso hacia dos zonas de tratamiento: etiquetado de aceitunas verdes o etiquetado de aceitunas negras.



Los bucles consisten en la repetición de determinada zona de tratamiento hasta que se verifique la condición impuesta. El cobrador admitirá pasajeros hasta que se completen las plazas disponibles.



La decisión múltiple es otra de las estructuras habituales en los programas. Un ejemplo ilustrativo de un proceso de decisión múltiple es el de selección y envasado de naranjas de distinto calibre.

encontramos con una señal de desviación obligatoria (salto incondicional), no nos quedará más remedio que coger tal desviación hasta el punto en el que se enlaza de nuevo con la vía principal.

2. *Salto condicional*

La estructura de un programa puede ser representada mediante un diagrama de flujo. En el mismo puede aparecer un símbolo de decisión que determina la ejecución u omisión de una parte del programa según se cumpla o no una condición impuesta al efecto.

Si queremos entrar en un cine tenemos que comprar la entrada (condición). Al llegar a la puerta pueden ocurrir dos cosas: que, en efecto, dispongamos de la entrada, con lo que podremos pasar al cumplir la condición impuesta, o que no dispongamos de entrada, lo que nos obligará a pasar por la taquilla para adquirirla y poder entrar.

Esta situación podría representarse como sigue:

SALTO SI CUMPLE
LA CONDICION

PASAR POR TAQUILLA
Y COMPRAR ENTRADA

ENTRAR EN LA SALA

Otra toma de decisión muy frecuente es la elección entre dos tratamientos de acuerdo con una condición dada; éste es otro método de salto condicional.

Veamos un nuevo ejemplo: suponga que tenemos mezcladas una cantidad determinada de aceitunas verdes y negras. Necesitamos separarlas (condición) para envasar y etiquetar las verdes (operación 2) y hacer lo mismo con las negras (operación 1). Una vez hecho esto se cargan en un camión para su venta. La figura ilustra perfectamente el proceso seguido. En forma de «instrucciones de programa» podríamos expresarlo como sigue:

SALTO CONDICIONAL SI LAS
ACEITUNAS SON VERDES

OPERACION 1

SALTO INCONDICIONAL

OPERACION 2

CARGAR EL CAMION

En definitiva, los saltos condicionales conducen a estructuras llamadas de *Decisión*.

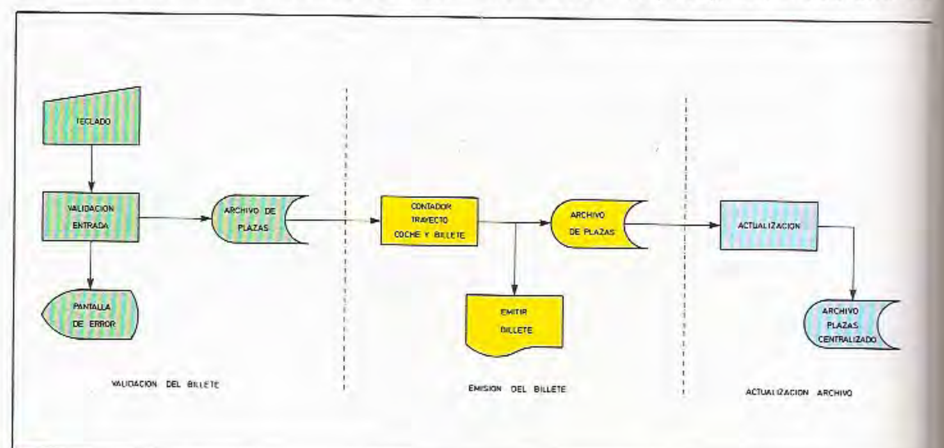
Los bucles

Consisten en la repetición continuada de una parte del programa hasta que se cumpla determinada condición.

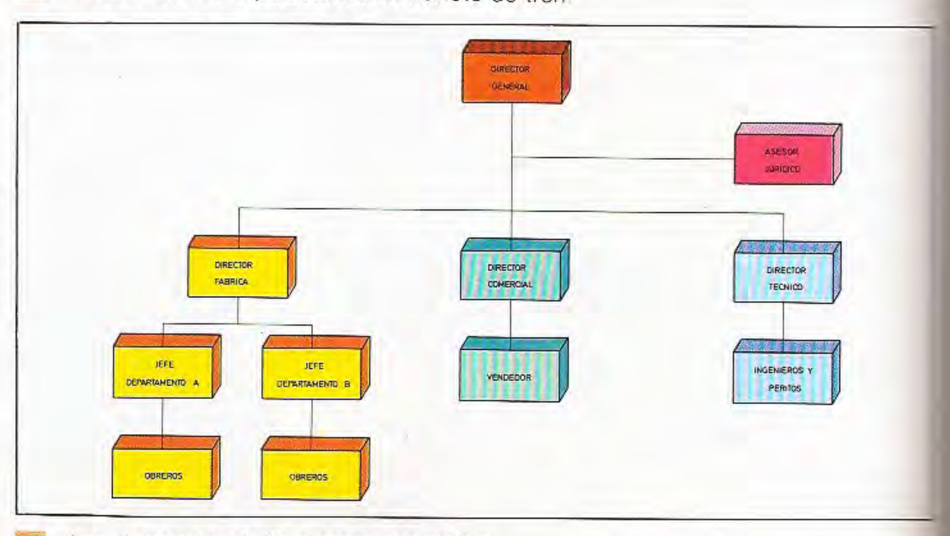
El caso más interesante es aquel en el que las repeticiones sucesivas se controlan mediante un índice, el cual se modifica en una cantidad constante en cada pasada, desde un valor inicial dado hasta que alcance un límite también prefijado.

El control de los valores del índice exige operaciones de asignación del valor inicial, de incremento y de comparación con el valor máximo. La primera operación debe realizarse necesariamente antes de entrar en el bucle, las demás se podrán alternar en cualquier orden.

De nuevo vamos a aclarar lo expuesto con un ejemplo. Suponga que tenemos un autobús de 70 plazas. Cuando empiezan a subir los pasajeros, el cobrador va



Los organigramas de sistemas permiten la representación gráfica de un tratamiento de información. En el ejemplo, se representa la secuencia de procesos parciales asociados a la expedición de un billete de tren.



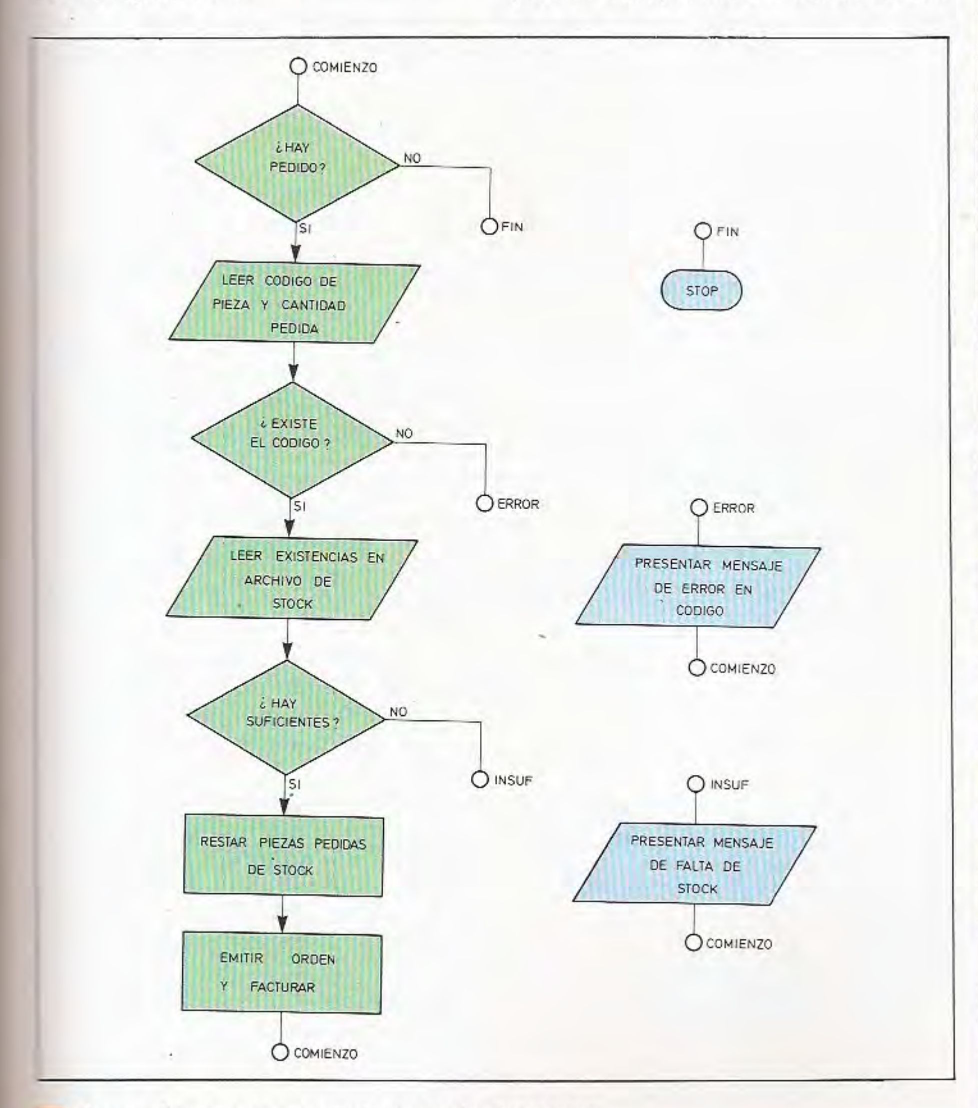
Los diagramas de flujo organizacionales o estáticos representan niveles de responsabilidad o jerarquía y la dependencia o relación entre ellos.

contándolos hasta que las 70 plazas queden cubiertas; a partir de ese momento, no autorizará la subida de ningún pasajero más.

Esta es una típica estructura de bucle. El índice está a cero cuando el autobús se encuentra vacío (ver figura); al entrar una persona, el índice aumenta en una unidad; a continuación, viene la pregunta ¿han subido ya los 70 pasajeros? (comparación con el valor máximo); la respuesta es evidentemente NO, con lo que se deja subir a un nuevo pasajero (índice = 2) y así sucesivamente. La salida del bucle se producirá cuando el índice sea igual a 70.

Decisión múltiple

Una tercera estructura de programa es la derivada de la técnica de decisiones múltiples, esto es: derivada de la posibilidad de elección de uno entre varios caminos. Su puesta en práctica precisa de un nuevo elemento llamado CONMUTADOR (K), al que se le asigna el número de la vía a seguir dentro del programa. El rombo que simboliza la decisión se sustituye por un triángulo. Un ejemplo de este tipo de estructura es el de envasar naranjas de diferentes tamaños que son transportadas a través de una



Los ordinogramas u organigramas de programas detallan los pasos que sintetizan un determinado proceso; por ejemplo, la actualización de piezas de un almacén.



Organigramas y ordinogramas

En general, podemos definir a un organigrama como una representación gráfica de un proceso, estructura organizada, etc.

Se utilizan en cualquier actividad de la vida cotidiana. Son de dos tipos: estáticos u organizacionales y dinámicos u operacionales.

Los estáticos u organizacionales representan los niveles de responsabilidad o jerarquía, así como los de dependencia entre las unidades o personas de una organización.

Los organigrmas dinámicos son representaciones gráficas del sistema de proceso de información y nos facilitan la labor de análisis y de entendimiento de los procesos.

Aunque existen diferentes tipos de organigramas, sólo nos referiremos a dos de ellos: los organigramas de sistemas y los organigramas de programa u «ORDINOGRAMAS».

Organigramas de sistemas

Permiten la representación gráfica de un proceso de datos, indicando las entradas y salidas de información con sus soportes y archivos, sin entrar en el detalle de cómo se realizan las operaciones. Se segmentan de forma que el flujo de información vaya de izquierda a derecha y de arriba a abajo.

Ordinogramas u organigramas de programas

Representan con detalle los pasos necesarios para realizar un proceso determinado. Sirven para ayudar al programador a realizar su trabajo. Mientras que el organigrama de sistemas daba más importancia a los medios y unidades, en el ordinograma se enfatizan los pasos necesarios para convertir los datos de entrada en información de salida. No se especifica el tipo de periférico que se utiliza, pero sí se especifica el archivo asociado a la operación.

cinta. Según sea el calibre de la naranja (asignación al conmutador del número de vía a seguir), las diversas compuertas se abren o permanecen cerradas.

Tablas de decisión

Las tablas de decisión permiten, en casos complejos, presentar de forma fácil las condiciones de un problema y la acción o acciones a adoptar en cada caso. Estas tablas, que se construyen a modo de *cuadros de decisión,* pueden sustituir a los ordinogramas en muchos casos.

La tabla se encuentra dividida en cuatro zonas, fraccionadas por una división horizontal y otra vertical.

En la parte superior izquierda se ponen las condiciones que hay que tener en cuenta para la resolución del problema. En la zona inferior izquierda se anotan las acciones a tomar en función de las condiciones. En la parte superior derecha se escriben las reglas o caminos alternativos con los símbolos S (sí), N (no), = (igual), ≠ (distinto), etc. Si el cuadro está en blanco indica que debe ser ignorado o que es irrelevante para el problema. En la zona inferior derecha se señalan con sendas «X» las acciones que hay que tomar.

Las normas a seguir pueden resumirse como sigue:

a) El juego de condiciones debe ser único.

b) Las acciones a ejecutar se realizan en el orden que están escritas. En caso contrario debe sustituirse la «X» por el número de orden de ejecución. Para ver cómo se organiza en la práctica una tabla de decisión vamos a analizar dos ejemplos. El primero de ellos coincide con la tabla de decisión correspondiente al caso de que al ir una mañana al trabajo queramos o no llevarnos el abrigo, según la temperatura ambiente. Los símbolos que se utilizan son SI, NO y «—» (que sustituye al espacio en blanco).

En el segundo ejemplo presentamos la problemática de expedición de billetes de avión en una oficina de una línea aérea. En este problema se trata de probar si hay plazas disponibles de la clase que desea el cliente y, en caso de falta de disponibilidad, averiguar si prefiere viajar en otra clase o apuntarse a la lista de espera.

Para terminar, recordemos que las tablas de decisión deben incluir todas las posibles decisiones que pueden presentarse dentro del proceso en cuestión.

TABLA DE DECISION DEL PROCESO DE USO DE ABRIGO

CONDICIONES	REGLAS		
	1	2	3
1 HACE FRIO	SI	NO	NO
2 PUEDE HACER FRIO	—	NO	SI
ACCIONES			
1 COGER EL ABRIGO	X	—	X
2 NO COGER EL ABRIGO	—	X	—

Tabla de decisión asociada al «uso del abrigo». Las condiciones establecidas y las acciones a adoptar se reducen en este caso a dos.

TABLA DE DECISION DE UNA AGENCIA DE VIAJES

CONDICIONES	REGLAS							
	1	2	3	4	5	6	7	8
1 Pide billete de 1.ª	S	S	S	S				
2 Pide billete turista					S	S	S	S
3 Hay plaza en 1.ª	S	N	N	N		S		N
4 Hay plaza en turista		S	N		S	N	N	N
5 No importa cambiar		S	S	N		S	N	S
ACCIONES								
1 Emitir billete de 1.ª	X					X		
2 Emitir billete de turista		X			X			
3 Resta una plaza de 1.ª	X					X		
4 Restar una plaza de turista		X			X			
5 Poner en lista espera de 1.ª			X	X				X
6 Poner en lista espera turista			X				X	X

El proceso de expedición de billetes en una agencia de viajes puede sintetizarse también en una tabla de decisión sustitutiva del correspondiente ordinograma.

Programación

Técnicas, documentación y rutinas de utilidad



La resolución de un problema por medios informáticos no es, habitualmente, una tarea trivial. No basta con sentarse y empezar a escribir líneas de instrucción. Hay que tener en cuenta que el problema no se resuelve realmente en el momento de la codificación, sino en las fases previas de análisis y diseño del programa.

Los dos pasos señalados son los más importantes y tal vez los más complejos de todo el proceso. Dependiendo de ellos se procederá a codificar el programa que aportará la buscada «solución informática».

Cuadernos de carga

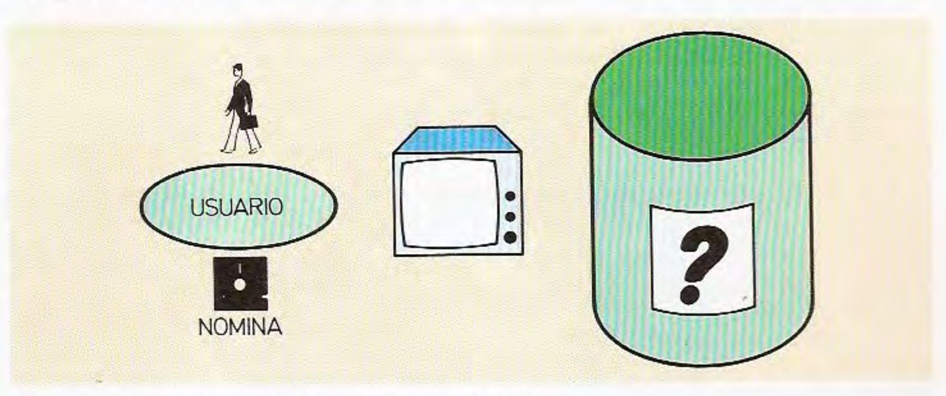
Una vez que los analistas y diseñadores consiguen tener una idea clara de lo que tiene que hacer el ordenador, empiezan a pensar la solución con ideas informáticas: ficheros, formas de acceso, cantidad, organización, flexibilidad, etc. Ellos son los que proporcionan a los programadores los llamados cuadernos de carga en los que está la descripción detallada de toda la aplicación, programa por programa.

En la descripción de los ficheros que intervienen se indican los métodos de organización de cada uno de ellos (secuencial, secuencial indexado, relativo, etc.) y si son parte de una base de datos que ya existe en ese momento. También se indica el método de acceso a estos ficheros (la estructura no es definitoria en cuanto al método de acceso a un fichero). En una base de datos esta opción es más importante y también más flexible.

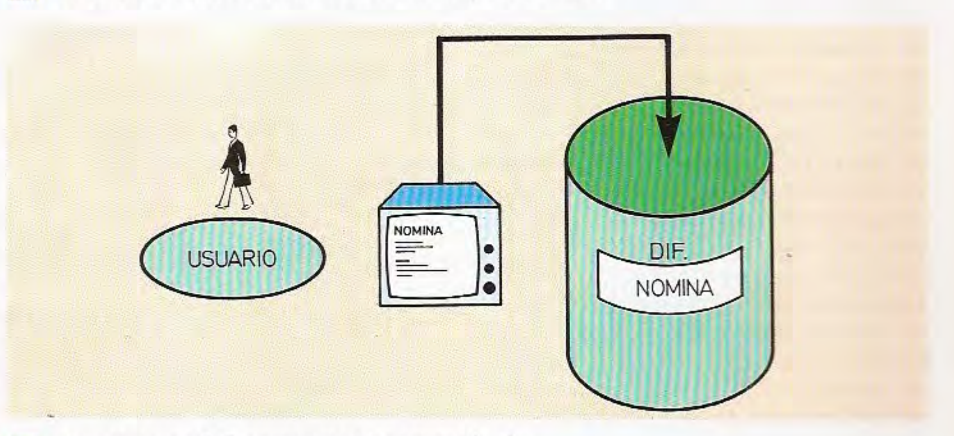
Dados los ficheros y sus características, se definen los datos que intervienen en cada uno de ellos, indicando su tipo y longitud. Se suele añadir un comentario del dato referido al conjunto de la aplicación, lo cual será de gran utilidad posteriormente en el momento del mantenimiento de los programas.

A efectos de comprobaciones posteriores se indican, si procede, los límites para cada uno de los campos de datos. Por ejemplo, será muy bien recibido por el oficinista un talón con su nómina mensual por valor de 1.500.000 pesetas. Al margen de pensar que por fin se reconocen sus méritos, lo más cabal es que imagine que se ha equivocado el ordenador. Para evitar este tipo de rrores, los programas deben comprobar que los datos estén siempre dentro de los márgenes correspondientes.

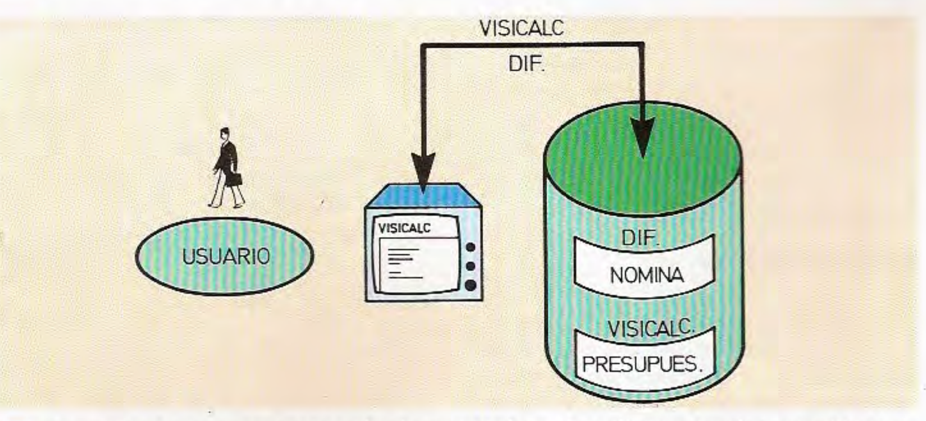
El siguiente paso consiste en describir



La solución de un problema utilizando la informática se obtiene fundamentalmente en la fase de análisis.



Los ficheros DIF permiten el intercambio de información entre dos o más programas.



La existencia de una aplicación de un interface de software pueden permitir a un empresario utilizar los datos allí contenidos para elaborar un presupuesto.

la forma en que los datos van a entrar y salir del proceso.

Pueden ser **procesos con resultado interno**, si son a nivel de fichero-fichero sin la intervención de usuarios finales, y **externos**, cuando la entrada o la salida tiene lugar hacia una persona normal y corriente (el operador). Estos casos son los más problemáticos.

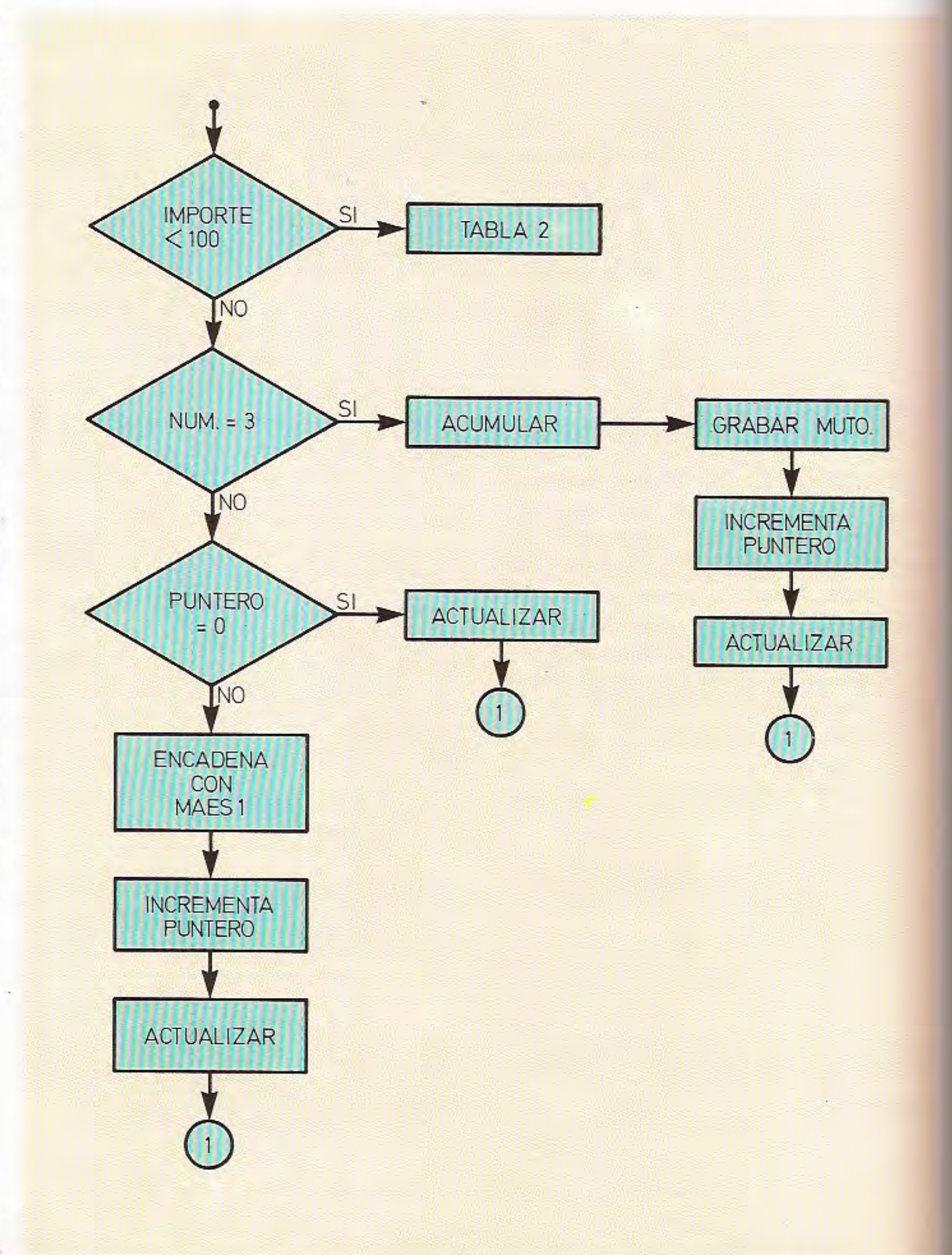
Las hojas de diseño de pantallas son las que el programador utiliza para confeccionar la entrada y salida de datos a través de este dispositivo de visualización. En ella se indican los campos a visualizar, su tipo, atributos... Un campo puede ser únicamente de salida y, por tanto, el usuario no podrá llegar a introducir ninguna información en esa zona (es sólo para uso del programa). Si es de entrada/salida, el operador podrá, cuando el programa lo permita, introducir la información correspondiente. También hay campos de entrada visible (lo que se teclea no se ve). El tratamiento de la información en la pantalla es una de las partes más complicadas y al mismo tiempo más bonitas. Un ejemplo de la importancia del tratamiento de la pantalla lo ofrece el ordenador LISA, de la compañía APPLE, o el STAR de XEROX (su predecesor), programas o entornos especiales como WINDOWS de Microsoft, etc.

La forma en que se obtendrán los resultados o datos por impresora se indica también en una hoja especial. Gracias a ella el programador podrá diseñar de forma mucho más sencilla los formatos de presentación en papel. En esta hoja se describen las posiciones de comienzo de los diversos campos y lo que ocupan; también se describe el formato de salida, es decir, la máscara con la que se imprimirá ese dato.

Descripción de los programas

Las descripciones de los programas se pueden hacer de forma natural: escribiendo los detalles oportunos, acompañados por diagramas de bloques del conjunto y cualquier otra aclaración que facilite al programador su tarea. También se pueden dar una serie de tablas de decisión que detallen los procesos más complicados. Estas tablas están formadas por cuatro cuadrantes. En ellas se enuncian los grupos de condiciones, los posibles estados, las acciones a ejecutar y la indicación de las acciones que se llevan a cabo para cada condición.

Se puede decir que son la representación en forma de tabla de un diagrama de flujo y tienen la ventaja de resultar más legibles. Cada posible condición sólo puede tomar dos valores; así, pueden resultar tablas muy grandes con la ventaja de tener todas las posibilidades contempladas. Existen métodos matemáticos para la reducción de la complejidad de dichas tablas.



La descripción del proceso de un programa puede hacerse, tanto en forma de diagrama de flujo, como de tablas de desición.

Comprobaciones

Cuando el programa ya está acabado hay que dedicar un tiempo, que nunca es el suficiente, a probarlo. En esta etapa aparecen toda una serie de fallos que normalmente no pasan de ser pequeños errores de programación. A partir de ese momento los programadores darán los últimos retoques y se implantará definitivamente el programa en el ordenador.

Simuladores de programa

Es un hecho frecuente que durante la elaboración de una aplicación algunas partes comunes no estén hechas cuando se las necesita. En el caso, por ejemplo, de las rutinas especiales de acceso a ficheros a través de programa. Estas rutinas pueden liberar al programa principal de muchos problemas y hacer que sea más legible.

Normalmente, las rutinas siempre tardan más en confeccionarse de lo que se tarda en llegar al punto del programa

TABLA DE DECISION (TABLA 1)

IMPORTE < 100	S	N	N	N
NUMERO = 3	X	S	N	N
PUNTERO = 0	X	X	N	S
IR A TABLA 2	•	—	—	—
ACUMULAR	—	•	—	—
GRABAR MOVIMIENTO	—	•	—	—
INCREM. PUNTERO	—	•	•	—
ACTUALIZAR	—	•	•	•
ENCADENA CON MAES 1	—	—	•	—
(1)	—	•	•	•

S = SE CUMPLE LA CONDICION

N = NO SE CUMPLE LA CONDICION

X = CONDICION INDIFERENTE

• = ACCION A EJECUTAR

— = ACCION A NO EJECUTAR

Las tablas de decisión constan del enunciado de las condiciones que deben cumplirse, de los estados que pueden tomar, de las acciones a ejecutar y de la indicación de las que se llevan a cabo para cada condición.



Una vez la aplicación terminada, es el momento de entregarla al usuario para las pruebas. En este momento empieza la discusión entre el resultado obtenido y lo que el usuario en realidad deseaba.

principal en el que es preciso usarlas. Se crean entonces los denominados simuladores de programa, cuya entrada será exactamente igual a la que se tendrá en el módulo definitivo y cuyos resultados serán una serie de constantes que facilitarán la progresión en el diseño del programa principal.

Utilidades para la programación

Para facilitar en todo lo posible el trabajo de los programadores de aplicaciones existen los denominados programas de utilidad.

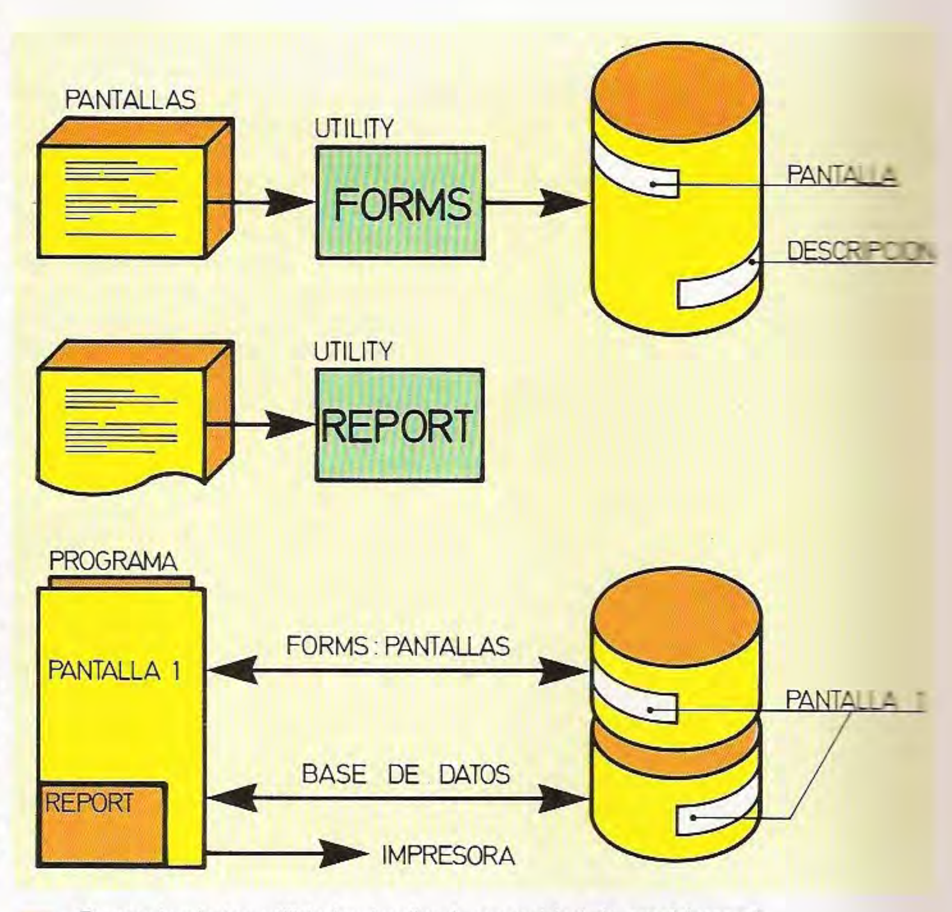
Supongamos que debemos crear una aplicación comercial que tendrá como usuarios finales a los empleados de unas oficinas. Debe ser, por tanto, un programa de tipo interactivo, de ejecución inmediata y de fácil utilización.

A la hora de diseñar una aplicación, aparte de toda la información necesaria para definirla, resultan de gran ayuda las denominadas «rutinas de utilidad».

Una primera necesidad es la creación de pantallas para la consulta de información. Esta suerte de «hojas» serán las que el usuario «rellenará» con los datos y en las que verá la información que necesita. A primera vista parece que es una cosa sencilla. Con las sentencias PRINT, LOCATE y unas pocas más ya tendríamos todo resuelto. Pero esta estructura es muy frágil. Los tiempos de dibujo del formato de pantalla son grandes. Además, estas sentencias son parte del programa y ello implica un aumento de la longitud del mismo. Pero el problema del «dibujo» de la pantalla no es el más grave. ¿Cómo indicaremos qué campos son de tipo numérico, alfanumérico, fechas, etc.? ¿Cómo especificar campos donde el usuario no «puede» escribir? ¿Y qué decir de las máscaras de edición?

Para que todas estas posibilidades queden contempladas será necesario indicarlas en el momento de la creación de la pantalla, de modo que cuando el programa utilice el formato no tenga que encargarse de controlar la mayoría de estas funciones.

Para resolver estos problemas existen utilidades especializadas en facilitar el diseño de pantallas.



El uso de ciertas utilidades implica la necesidad de amoldarse al análisis previo, debiendo los diseños de las pantallas, los listados y el esquema de la base de datos quedar definidos.

Los listados

Problemas parecidos se presentan con los listados. Existen para ello utilidades específicas para apoyar el diseño de formatos impresos. Una utilidad de esta categoría suele constar de tres zonas: una de descripción, otra de ejecución y la última de información.

En la parte de descripción se recoge lo que tiene que hacer el programa cada vez que un listado llega al fin de hoja; se define también el texto de la cabecera de página, la numeración de las páginas, la distribución y contenido de los pies de página...

Con esta herramienta la creación de un listado no presenta grandes problemas; permite una gran flexibilidad en el momento del diseño y además las mo-

Utilidades para definición de pantallas

Partiendo de la base de que la mayor comunicación con los usuarios finales se efectúa a través de la pantalla y el teclado, se deben dedicar los mayores esfuerzos a desarrollar un buen «interface» entre ambos.

Veamos algunas cuestiones que debe resolver por sí misma una utilidad de esta índole, sin que el programa que la utilice deba incorporar un tratamiento especial.

Lo primero es la existencia de por lo menos tres tipos de campos: salida, entrada/salida y entrada/salida controlada (el tipo de sólo entrada es secundario, su función ya la realiza el tipo de entrada/salida).

A los campos de salida el usuario no puede acceder, sólo el programa. En el caso de entrada/salida se permitirá que el usuario y el programa tengan acceso a esa zona. El tipo de entrada/salida controlada está muy relacionado con el orden en el que el cursor se traslade a cada campo de pantalla.

Cuando se defina el formato se indicará, además del tipo de campo, cuál es el campo siguiente y cuál el anterior, de forma que cuando se pulse RETURN se sepa a qué campo debe saltar el cursor. Los tipos de campo fecha, numérico y alfanumérico, deben quedar plenamente caracterizados, de tal forma que no permitan teclear una letra en un campo numérico.

Después de introducir una cantidad numérica y pulsar RETURN, ésta debe quedar ajustada a la derecha, y si tiene máscara definida, debe visualizarse el dato en consecuencia.

Los atributos de cada campo deben ser independientes como grupo. Así, todos los de salida no tienen por qué aparecer subrayados, ni todos los de entrada/salida en visualización realzada. Según las necesidades deben admitir diferentes atributos.

dificaciones son muy sencillas. Evidentemente, esta utilidad puede ser complementaria de otra de tipo SPOOL que gestione las prioridades para la salida por impresora («colas»).

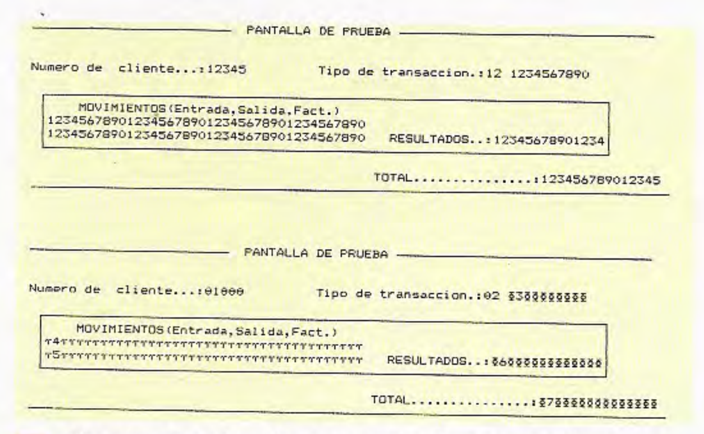
Clasificación

Algunas veces es necesario que la información se organice de forma distinta a como está en ese momento. Podría ser interesante, tal vez, disponer de un listado ordenado por uno de los campos secundarios o por varios.

El método para ello suele ser identificar los ficheros que van a intervenir en la ordenación y el fichero resultado de la ordenación. Los campos que clasificará el ordenador pueden pertenecer a un grupo de ficheros de entrada. Desde luego, se indicará también el orden en que se clasificarán: ascendente o descendente. Este proceso se suele realizar mediante una utilidad denominada SORT.

Cuando se efectúa un SORT se crea un fichero de salida, el cual almacena el resultado del proceso. Dentro de la propia utilidad puede existir una parte que crea este fichero vacío, dependiendo de las longitudes de los campos a ordenar, longitud del registro ordenado, cantidad de registros a clasificar, etc.

La potencia de esta utilidad es notable; se emplea cuando en el diseño de la aplicación se asume este método para casos no críticos y de realización ocasional. En caso contrario se pensarían métodos alternativos que garantizaran un estado de ordenación permanente.



Ejemplo de definición de pantalla. En la parte inferior tenemos la pantalla tal como queríamos que resultara en nuestro programa. Los números indican la longitud del campo a efectos de ayuda.

```
10 REM .............. PRUEBA DE CROS REFERENCE .....................
20 Nombre$="PRUEBA de Programa"
30 FOR Indice=1 TO 20
40  Linea=Indice
50  Columna=Indice+1
60  GOSUB Saca
70 NEXT Indice
80 END
90 Saca:! ......... saca mensaje Nombre$ en linea y columna ...........
100 CURSOR (Linea,Columna)
110 DISP Nombre$
120 RETURN



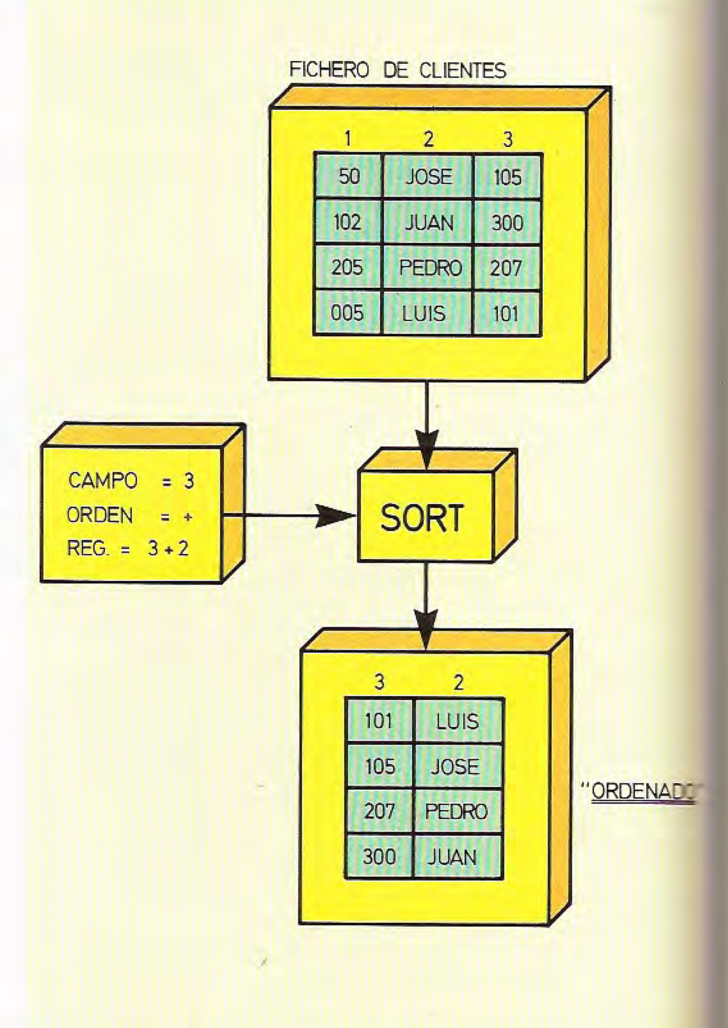
 HP 250 Cross Reference Table.              File: JUAN


 PROGRAM: JUAN            (LINES 10-120)

 1                  CON          30    50
 20                 CON          30
 Cloumna            R          ***UNREFERENCED***
 Columna            R          ***REFERENCED, BUT UNDECLARED    50   100
 Indice             R          ***REFERENCED, BUT UNDECLARED    30    40
                                       50    70
 L                  R          ***UNREFERENCED***
 Linea              R          ***REFERENCED, BUT UNDECLARED    40   100
 Nombre             $          ***REFERENCED, BUT UNDECLARED    20   110
 Saca               LABEL        -90    60

     *** End Of Cross Reference ***
```

Ejemplo de programa de referencias cruzadas. La variable «CLOUMNA» no existe en el programa como tal, pero durante el desarrollo existió. Para eliminarla se utilizarán las instrucciones SAVE y GET.

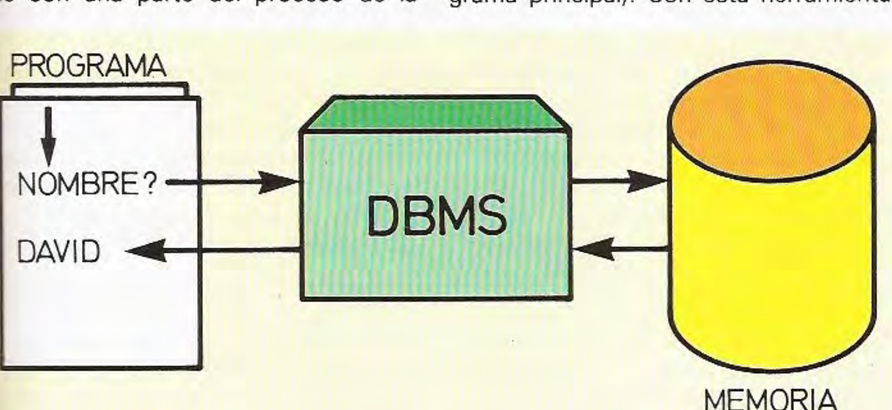


Cuando se efectúa un SORT se forma un «ordenado» a partir de un fichero primitivo. En el ejemplo de la figura se han ordenado los clientes en función del campo 3 y en orden creciente.

La detección de errores

Cuando se acaba un programa aparecen, inevitablemente, una serie de errores. Los métodos para su detección son de muchos tipos y, en todo caso, la habilidad del programador es importante para encontrar los fallos.

Un posible método es la creación de las llamadas referencias cruzadas (CROSS REFERENCE). En general, todos los compiladores las proporcionan, ya que son una parte del proceso de la compilación. La información que dan es variada. En primer lugar, todas las variables que existen en el programa (ordenadas alfabéticamente) y su tipo (real, entera, alfanumérica, matriz y dimensiones). También todas las constantes existentes y la línea en que aparecen. También las etiquetas y las funciones definidas por el usuario aparecen con las indicaciones de donde se declaran y donde se referencian. Los subprogramas tienen un tratamiento igual que el programa que lo llama (depende del sistema la posibilidad de estos subprogramas y su forma de relación con el programa principal). Con esta herramienta



La gestión de una base de datos puede considerarse más como un subsistema que como una unidad. Su función consiste en relacionar el programa con los datos y su estructura (DBMS = Data Base Management Subsystem).

```
* * * * *   TABLA DE RELACIONES DE CAMPOS   -----------------------------------
```

Numero	Col.	Lin.	Tipo	Sig.	Ant.	Fill	Modo	MASCARAS - LONGITUD
1	23	4	4	2	1	1	3	\9.999
2	58	4	4	4	1	1	3	2
3	61	4	2	4	2	2	2	10
4	3	8	1	5	2	0	1	40
5	3	9	1	5	4	0	1	40
6	59	9	4		0	0	2	\99.999.999.999
7	65	12	4	0	0	0	2	\999.999.999.999
8								
9								

```
Tipo:1.-Alfa 2.-Fecha 3.-Mayu. 4.-Num.   Modo:1.-I/O 2.-Output 3.-I/O Cont.
-Segmento de la pantalla que contiene el campo a modificar.

        S Siguiente Bloque          M Modificar
```

Esta tabla indica cada una de las especificaciones para los distintos campos. Esto permite distinguir entre campos numéricos, alfanuméricos y fechas. Asimismo, tiene incluida la posibilidad de máscara y de solapar varios campos en la pantalla.

Para saber más

¿Qué son las máscaras?

La máscara es una especie de filtro por el que los datos deben salir. Lo hacen según la forma que ésta tenga. Las máscaras no son sólo relativas a la edición en pantalla o impresora, también hay máscaras de selección en las interrogaciones a una base de datos o a un archivo; en ellas se indican las condiciones que debe cumplir una información para que se considere seleccionada.

¿Qué es un interface de software?

Cuando un programa debe transmitir información a otro, lo puede hacer de varias formas: a través de ficheros, de variables en zonas protegidas, etc. El interface entre dos o más programas supone la integración de los intercambios entre ellos. Existen, por ejemplo, los denominados ficheros DIF (Data Interchange File), cuyo formato estándar hace posible que puedan ser leídos y tratados por diversas aplicaciones.

¿Qué es una «ruptura» en un listado?

Cuando se pretende, por ejemplo, sacar un listado por provincias y dentro de éstas por poblaciones, cada vez que se lee un dato y se comprueba que la población ya no es igual que la anterior existe un primer nivel de ruptura. Cuando lo que cambia es la provincia, el nivel de ruptura es el mayor y lleva implícito una ruptura de todos los inferiores. En cada ruptura puede alterarse la acción a ejecutar.



Herramientas para el desarrollo de software

En el momento en que se le plantea a un programador el desarrollo de su primera aplicación de tipo «profesional», marca un ascenso de categoría en el ranking de la programación.

Empieza la codificación sin ningún tipo de reparo. Cuando tiene escritas ya una buena cantidad de líneas, cae en la cuenta de que para poder ver las líneas que están por encima de la pantalla visible debe utilizar el comando LIST con el intervalo que se quiere ver.

Evidentemente se trata de una incomodidad y desearía tener como herramienta de diseño un «scrolling» de la pantalla en todos los sentidos. Y no digamos nada de la edición orientada a la línea. Cada modificación significa «llamar» a la línea, dar una serie de especificaciones de cambios y otras cosas. Es preferible teclearla entera de nuevo. Superados estos primeros traumas, observa que debe renumerar el programa para permitir la inserción de una nueva línea que ya no cabe donde debía ir y, desgraciadamente, no existe en su editor el comando de renumeración.

Sucede a menudo que se tiene una variable por algún sitio y hace falta cambiarle el tipo. Pero resulta que no se cuenta con el comando de búsqueda a través de todo el programa y, por supuesto, tampoco con un editor con el que se pueda buscar a través del texto del programa en ASCII y operar un cambio automático todas las veces que sea necesario.

Suele ser corriente la necesidad de fusionar dos programas; para ello, la única forma es introducir una serie de instrucciones POKE a posiciones que nos contó un «amigo» y esperar que funcione la carga sin borrar el programa existente. Después, más instrucciones POKE y programa unido.

Un comando un tanto raro puede ser el que indente. Es decir, cree entradas más a la izquierda para las instrucciones que están dentro de bucles FOR... NEXT, LOOP, WHILE, etc.

Desesperado de su «profesionalidad», el programador llega al momento de las pruebas y no cuenta con ningún tipo de trazador o debugger. Debe introducir instrucciones de visualización por doquier para saber por dónde «va» el programa, así como instrucciones STOP, y pedir el contenido de variables, etc. Tras una experiencia de esta índole, no es preciso reiterar al afectado la importancia que revisten las utilidades para desarrollo de software.

cualquier modificación se realiza de forma más sencilla y más segura.

Debugger

Otra utilidad para la depuración de programas es el DEBUGGER o TRACE. Existen muchos tipos de rutinas de TRACE. Por ejemplo, las hay que exclusivamente muestran el número de línea por la que el programa pasa. Otras indican en la pantalla la sentencia en ejecución y aguardan a que se pulse una tecla para continuar, pudiéndose parar para examinar el contenido de una variable y seguir. También las hay que indican sólo las líneas de donde y a donde se realiza una bifurcación. Otra nos señala la línea en la que una variable dada cambia de valor.

En resumen, cuantas más herramientas se proporcione a los programadores, más rápidos y más seguros resultarán los programas.

Programación modular y estructurada

En busca de programas flexibles y transportables



Se puede decir que la programación modular es el sentido común aplicado a la resolución de un problema informático. Su objetivo es programar en base a módulos; cada uno de ellos representativo de una sección de programa que resuelve una parte del problema.

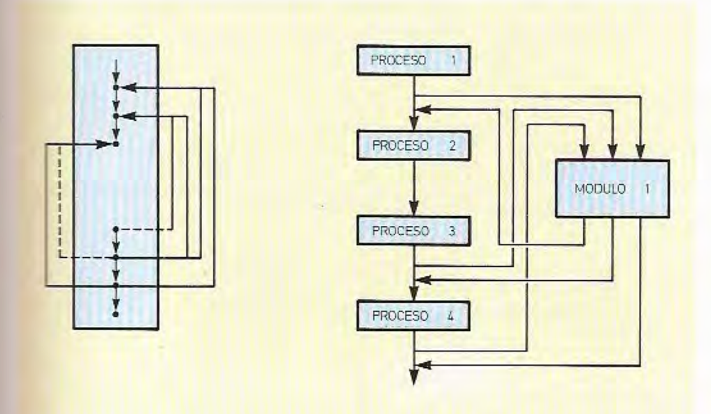
Cuando en un centro de cálculo, o a inferior nivel en nuestro ordenador personal, se necesita crear «una aplicación», el primer objetivo es definir el resultado final que se pretende obtener.

Modificaciones en un programa no modular

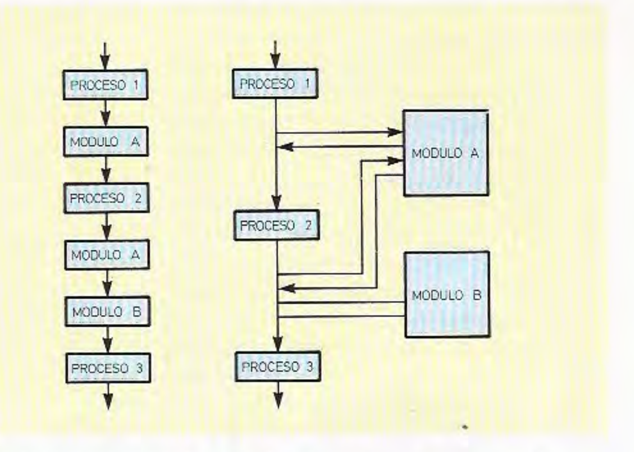
Con el objetivo final en mente, ya se puede empezar la codificación del programa (en el lenguaje que se haya considerado más necesario).

Después de codificarlo y de probar con muchos datos que el programa no contiene errores y gestiona perfectamente los periféricos, puede suceder que varíen los datos de partida del programa. Tristemente, el analista se da cuenta de que su trabajo anterior ya no sirve: tiene entonces que modificarlo. Si no ha tomado precauciones, lo más probable es que no localice la parte afectada por estos cambios. Y cuando la localiza, comprueba en ocasiones que un cambio de estas líneas del programa altera resultados intermedios empleados más adelante.

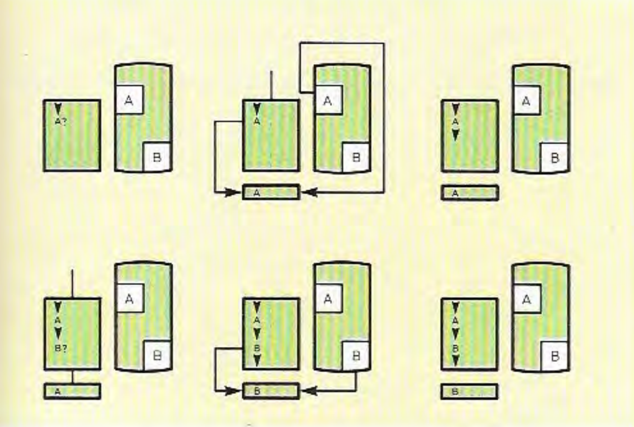
No cabe duda que si cada cosa estuviera en su sitio concreto, tendría mucho avanzado cuando se planteara la necesidad de efectuar algún cambio. Si, además, todas las partes tuvieran una función muy delimitada, la modificación resultaría relativamente sencilla. Sabría dónde, cómo y por qué efectuar los



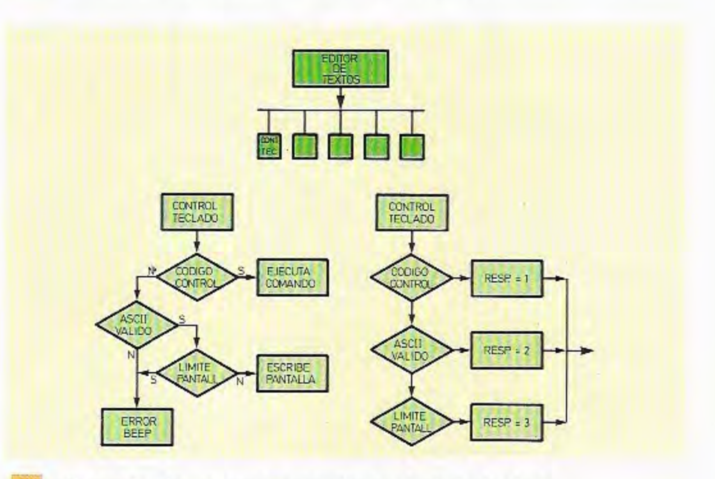
Ejemplo de organigrama de un programa no modular (a la izquierda), y de otro modular (a la derecha). En los programas no modulares las sentencias de salto pueden efectuarse desde cualquier parte del programa.



Un programa modular puede verse como una sucesión encadenada de módulos que se ejecutan uno tras otro, o como un conjunto de llamadas a subrutinas cerradas, y contenidas en cada uno de los módulos.



Cuando un programa llama a un módulo no almacenado en la memoria principal, el sistema lo trae del disco, lo graba en las posiciones correspondientes y le cede el control de la CPU.



Ejemplo de un programa de control de teclado realizado modularmente. Este programa es un módulo utilizado en una aplicación cualquiera de procesamiento de textos.

cambios necesarios. Estaría haciendo una programación modular.

Objetivos de un programa modular

Los objetivos buscados con la programación modular son dos:

- Flexibilidad en los programas.
- Generalidad.

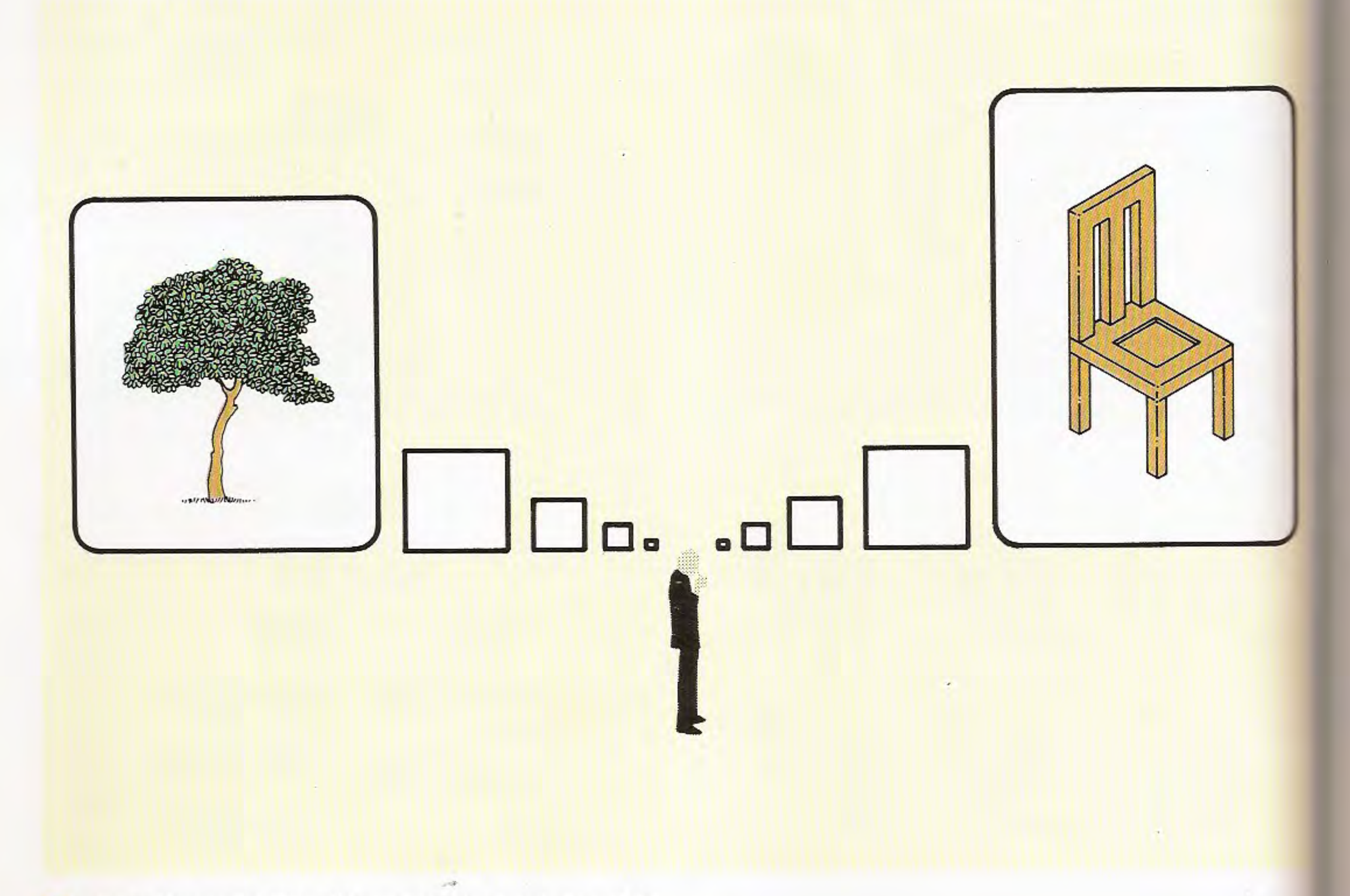
Flexibilidad: Cuando los programas no son lo suficientemente «adaptables» y aparece un cambio exterior que afecta a sus datos de partida, lo normal es que se vuelvan a rehacer o, sencillamente, que se descarten los programas. Debe ser posible, sin embargo, que el programa no sufra cambios profundos en la estructura y, muchísimo menos, en los datos internos. Una ley muy conocida en informática, y a tener siempre en cuenta, dice que todo sistema cambia con el tiempo y que nunca se llega a delimitar hasta que se pretende mecanizarlo.

Generalidad: Un programa diseñado para cumplir con una serie de objetivos muy abiertos nunca utiliza la totalidad de las funciones previstas para ejecutar una aplicación concreta. El usuario está pagando entonces por algo que no está usando.

Puede llegar un momento en que ambos términos, flexibilidad y generalidad, se solapen y proporcionen así un resultado mucho más «interesante», tanto para el usuario como para el programador.

Programas transportables

Aparece entonces un nuevo concepto: el de *transportabilidad.* Se puede considerar a esta característica como algo supra-programático. Y, realmente, es una de las máximas de todo el departamento de análisis informático.



Los problemas no informáticos se pueden resolver también de forma modular. Un ebanista con sentido común, por ejemplo, fabrica sus muebles de acuerdo a reglas generales.

El concepto de transportabilidad se aplica a cualquier programa que puede ejecutarse en un entorno para el que no fue creado ni pensado.

Se pueden considerar dos formas de transportabilidad:

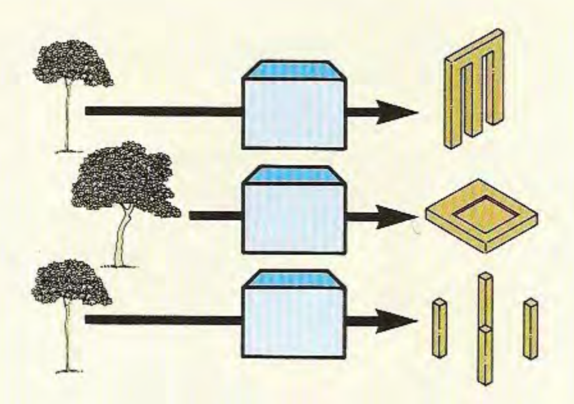
- Transportabilidad entre máquinas.
- Transportabilidad entre programas.

La transportabilidad entre máquinas se da cuando un programa que funciona en un sistema determinado se ejecuta en otro distinto. Esta es una característica para la que se dan menos soluciones en el momento del diseño.

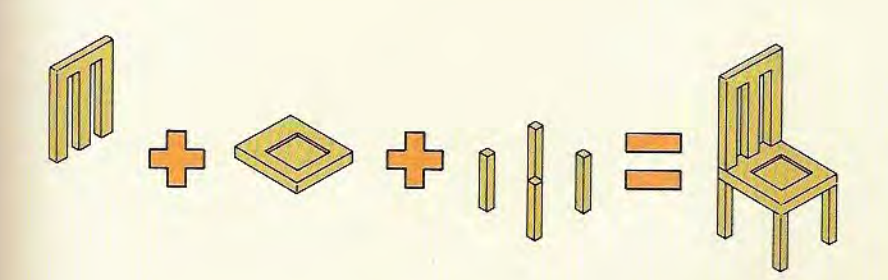
Mucho más normal es prever el intercambio entre aplicaciones dentro del mismo ámbito de la empresa o de la organización de una macroaplicación. Por ejemplo, si un programa necesita conocer todos los individuos que cumplen una condición determinada, se creará un «módulo» que haga «eso y sólo eso». Cuando para otra aplicación se necesite conocer los nombres de estas personas no será preciso volver a escribir el programa: se intercalará directamente este módulo en la correspondiente zona de la aplicación.

Características de los módulos

La razón última de que un programa sea modular es que al subdividir un pro-



Una silla se compone de cuatro patas iguales, del asiento, del respaldo, etc. Cada una de estas partes son los «módulos» de la silla.



Si se unen cada uno de los módulos, en la forma y el orden adecuado, se obtiene el mueble deseado.

Asignación de memoria

La administración de la memoria es una de las funciones más importantes de un sistema operativo. Sin duda, para aumentar el rendimiento de la ejecución múltiple de programas es necesario un control eficaz y flexible del espacio de memoria direccionable por el sistema.
El sistema operativo de un ordenador de tiempo compartido debe controlar que ninguna tarea entre en las zonas de memoria de los demás o del propio sistema. A pesar de todo, pueden existir zonas de memoria contiguas.
Una técnica aplicada en estas situaciones es la partición reasignable de memoria o la paginación mediante la participación reasignable. Cuando una tarea finaliza, se cubre la parte libre de memoria con las tareas que aún están pendientes o en curso de proceso. Esto obliga a que las direcciones absolutas del código objeto cambien para adaptarse a las nuevas posiciones. En este caso todas las direcciones son relativas a una determinada: la del registro de reasignación. Las nuevas direcciones se hallan sumando la dirección última a una cantidad de desplazamiento almacenada en el registro de reasignación.
La asignación paginada crea un espacio direccionable de cierta longitud, llamado página. La memoria física se divide en bloques de la misma longitud que la de la página. Cualquier página puede almacenarse, por tanto, en cualquier bloque. Cada uno de estos bloques tiene un índice para el cálculo de la dirección física.
Cualquier tarea se divide en un número de páginas, a cada una de las cuales se asigna un bloque. La dirección se calcula, en este caso, con el índice de cada uno de los bloques.
Paginación por demanda es otro método de asignación de memoria, que también se conoce por el nombre de memoria virtual. En este caso el espacio direccionable por los programas es generalmente mayor que la cantidad de memoria física disponible.

blema en varias tareas más pequeñas, directas y sencillas es más fácil de escribir, leer y mantener su codificación.

Cada módulo debe ser fácilmente visible e identificable. Otro programador debe ser capaz de encontrar, entender y modificar cualquier módulo del programa. Un módulo debe ser independiente e integrarse sin ajustes a la estructura más o menos jerárquica del programa principal.

Cabe afirmar, pues, que los módulos son zonas con una lógica propia, utilizables por un programa principal y que poseen una significación por sí mismas. La entrada y salida a los módulos deben estar bien definidas y ser únicas.

Los módulos se asemejan a las ramas de un árbol, independientes entre sí pero que a la vez están todas ellas unidas y relacionadas por el tronco. Este elemento director debe ser el algortimo de descomposición del problema en los módulos requeridos.

El algoritmo debe estar realizado para que sea fácil programar modularmente y, además, siguiendo los criterios de *transportabilidad* y *flexibilidad* del conjunto.

La modularidad está determinada por la elaboración de los algoritmos y por los análisis previos al diseño del sistema. El grado de modularidad es función de los conocimientos de informática y de la experiencia del programador ante estos problemas. Por ejemplo, la organización de un sistema operativo o de un compilador será tanto más flexible cuanto menor sea el número de parámetros fijos. Casi todos estos parámetros son contenidos de direcciones que dependen de la configuración existente en ese momento.

Condiciones de modularidad

Para que se pueda realizar programación modular es necesario que exista un soporte tanto de hardware como de software adecuado.

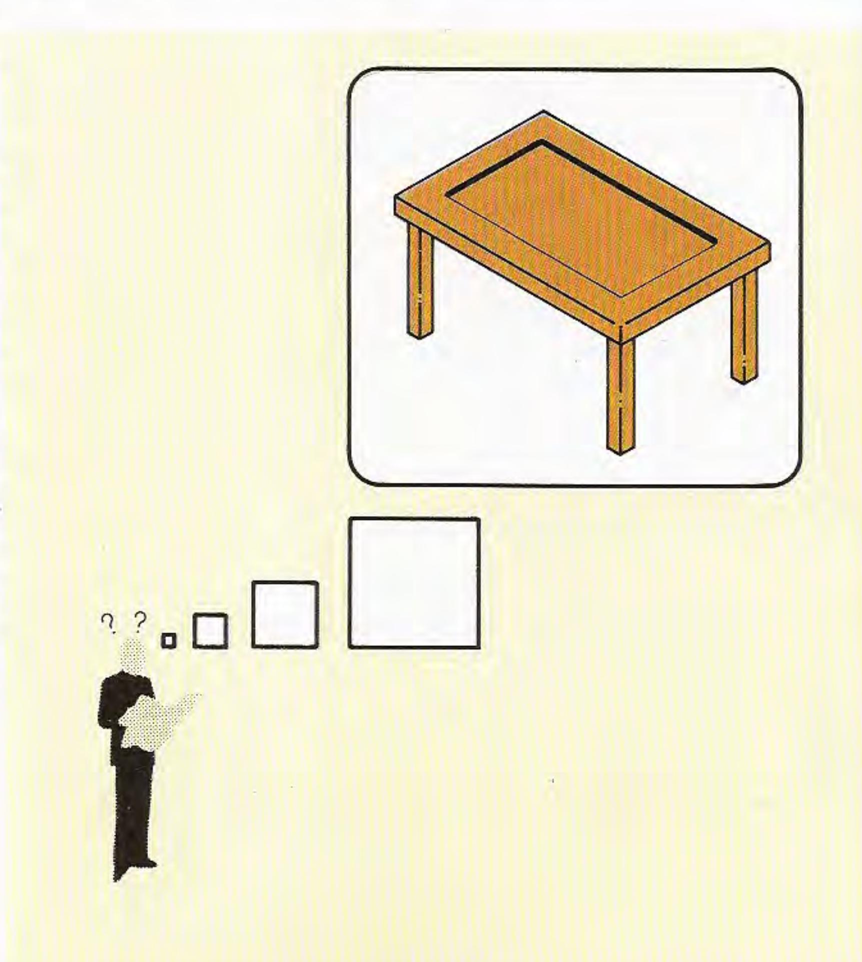
El hardware es preciso porque es impensable programar modularmente sin soportes de acceso directo y controladores independientes del procesador central. La responsabilidad recae, en este caso, sobre el software de base: sistema operativo, montador de enlace, reubicadores, etc.

El lenguaje, por su parte, debe contar con sentencias que permitan la existencia de este tipo de estructuras modulares en el programa. La sentencia más simple para escribir programas modulares es: DEF Función. Esta instrucción varía de unos sistemas a otros, aunque, normalmente, ocupa sólo una línea de programa en la que se escriben los parámetros que particularizan la función.

En otros equipos, para ejecutar una función concreta se salta a un punto del programa usando variables propias de esta función. Esta operación puede ocupar varias líneas de programa. El retorno al programa principal puede hacerse desde cualquier punto del módulo, devolviendo o no variables.

Instrucciones no tan potentes o menos flexibles que DEF Función son, por ejemplo, GOSUB, CALL, USR, SYS, etc.

Lo ideal es que la máquina trabaje con un lenguaje dotado de estructuras que permitan la implantación sencilla de módulos y su interacción. Estos lenguajes son los llamados *estructurados*. Su principal característica es que permiten una gran variedad de modos de segmentar, modular o encadenar otros módulos existentes.



Cuando el ebanista desea fabricar una mesa puede recurrir a algunos de los módulos creados para la silla: las patas, por ejemplo.

Programación modular y programación estructurada

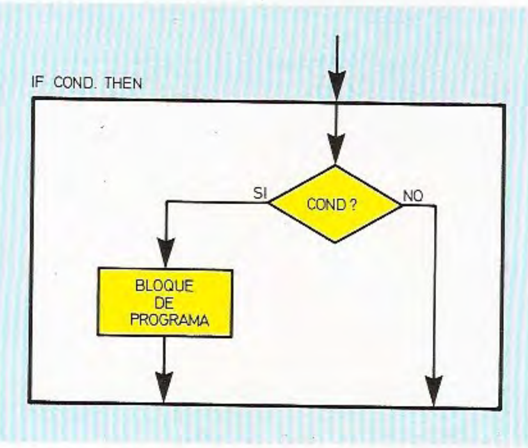
La experiencia ha demostrado que un programa que presente dificultades para ser modificado está condenado a la «muerte informática». Debe procurarse, por tanto, que los programas sean a la vez flexibles y transportables: flexibles para que se adapten con facilidad a cualquier cambio; transportables de forma que cualquier nuevo proceso pueda utilizar sus subrutinas sin introducir grandes cambios.

Conviene, para ello, emplear técnicas de programación que faciliten el desarrollo de software fácilmente modificable. De esta forma, el programador que utilice el software desarrollado anteriormente no tendrá que efectuar dos tareas muy tediosas:

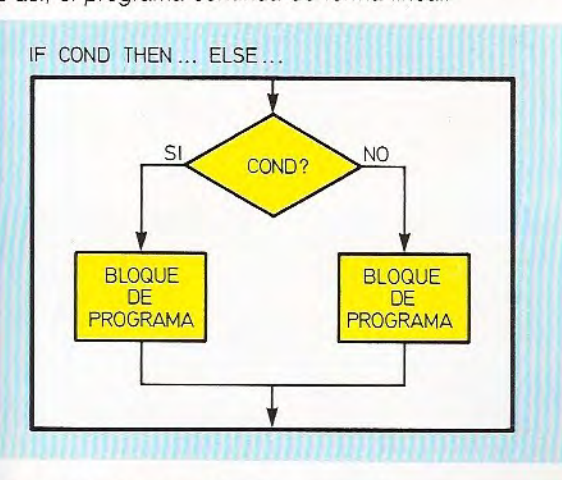
- Reescribir zonas del programa ya escritas.
- Probar subrutinas ya probadas.

Todas estas consideraciones se acercan a la idea de programación modular: cada problema debe descomponerse en una serie de problemas más pequeños hasta llegar a un nivel en el que cada uno de ellos no pueda reducirse más. En ese momento se ha llegado al escalón más bajo del análisis. Es entonces cuando realmente se puede resolver el problema planteado al principio. Cada uno de estos problemas mínimos realiza una sola función; de esta forma un problema de orden superior puede usar para su resolución «problemas mínimos» comunes a varios niveles.

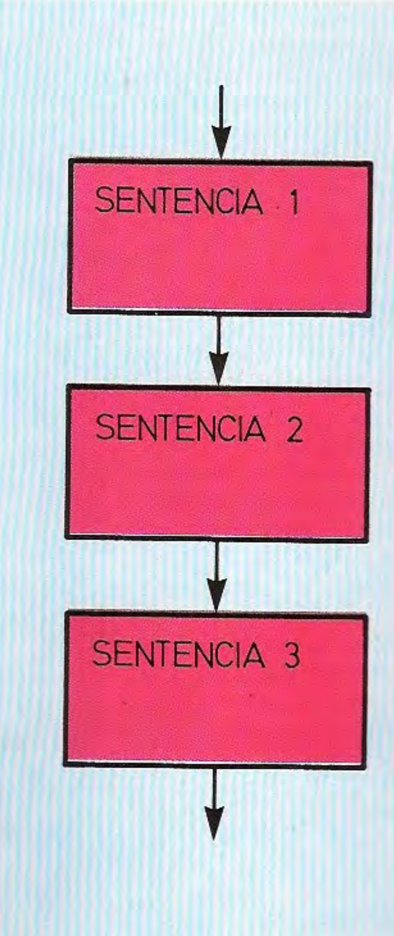
Una vez demostrada la necesidad de descomponer un problema general en «problemas mínimos», resulta obvio que éstos no son sino los módulos de que consta el programa. De esta forma se está haciendo a la vez programación modular y programación estructurada: el software obtenido es modular, mientras que las técnicas empleadas para desarrollarlo son estructuradas.



Bifurcación utilizada en programación estructurada. Si la condición se cumple se ejecuta el siguiente bloque; si no es así, el programa continúa de forma lineal.



Bifurcación del tipo «Si... Entonces... Sino...». De cumplirse la condición se ejecuta el bloque de la izquierda, en caso contrario, el de la derecha.



Bloque lineal. La principal característica de este tipo de estructuras es que contiene una entrada y una única salida.

Características de un programa estructurado

El desarrollo de programas modulares requiere un soporte software adecuado; ciertamente, el grado de modularidad obtenido depende del intérprete o del compilador empleados. En este sentido, resulta muy útil contar con instrucciones flexibles para la ejecución de subprogramación o módulos.

Para conseguir que los programas sean transportables es necesario programar en base a módulos de pequeño tamaño, cada uno de los cuales debe facilitar toda la documentación posible sobre su funcionamiento. Es preciso que con un simple vistazo al listado de cada módulo cualquier programador comprenda su funcionamiento.

Por otra parte, los algoritmos de un programa estructurado deben ser muy sencillos. Es preferible utilizar varias instrucciones separadas y visibles que una sola con muchos niveles de paréntesis, operaciones complicadas, etc. Una sentencia con cinco o seis instrucciones de tratamiento de cadenas, o con varias funciones definidas por el usuario, puede provocar el desconcierto en cualquier programador que intente comprender su funcionamiento.

Otro factor muy importante que determina la legibilidad de un módulo es la linealidad de la secuencia de sus instrucciones. En este sentido no resulta aconsejable el uso de sentencias «GOTO», pues cada vez que un programador se encuentra con una de ellas tiene que reconstruir mentalmente el organigrama del programa.

Si los módulos de un programa estructurado están bien construidos, cada uno de ellos ejecutará una sola tarea y no efectuará ningún tipo de saltos a puntos alejados del programa.

Tipos de sentencias de un programa estructurado

Varios autores han demostrado que cualquier programa estructurado puede construirse por medio de tres tipos básicos de estructuras. Estas son:

- Secuencia lineal.
- Bifurcación.
- Repetición.

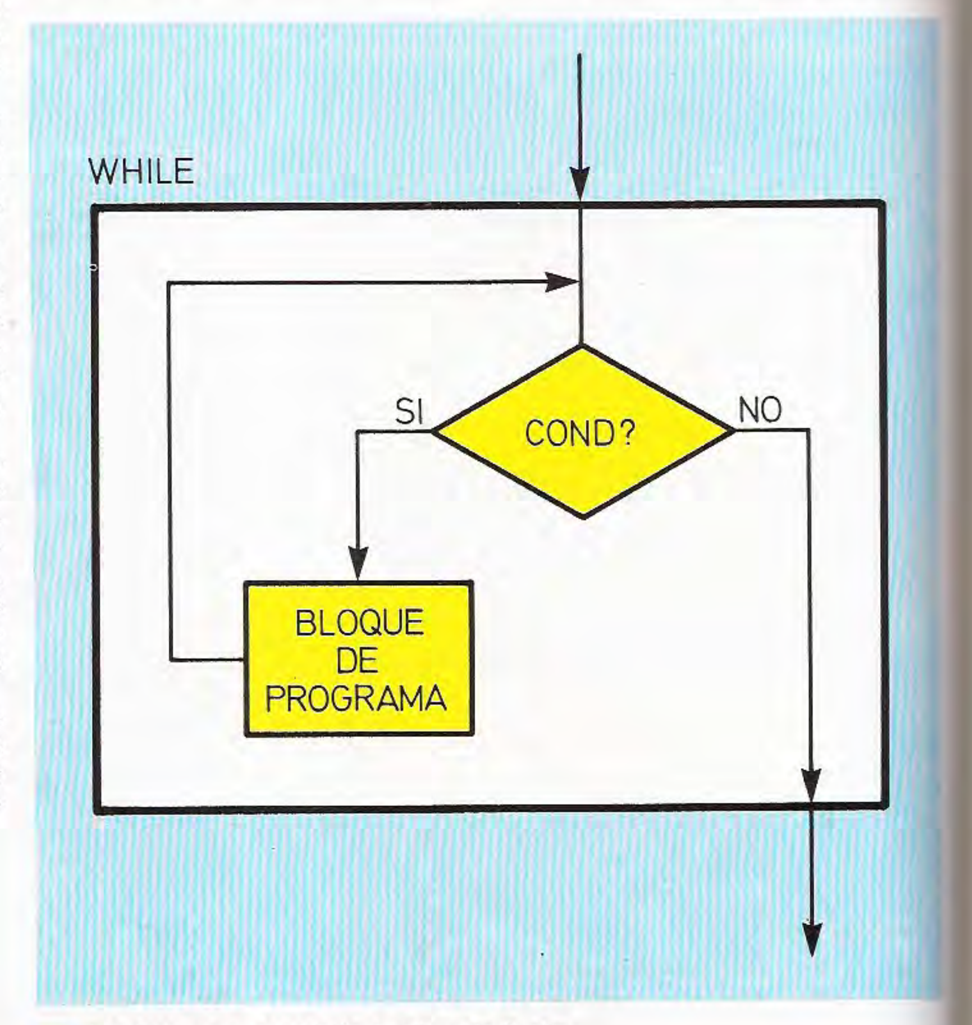
Las sentencias lineales son las más comunes en un programa estructurado. Representan una operación o acción ejecutada dentro del programa.

La bifurcación es la operación por la cual el ordenador escoge la acción a ejecutar dentro de un conjunto de posibilidades. Esta elección está determinada por el valor que tomen determinadas variables, calculadas anteriormente por medio de sentencias lineales.

Cuando se repiten varias operaciones hasta que una variable cualquiera tome un valor determinado, se está ejecutando una sentencia de repetición.

Aunque con estos tres tipos de sentencias se puede construir cualquier programa estructurado, no queda garantizada su legibilidad. Para asegurarla se crean secuencias lineales independientes.

Cuando un bloque de instrucciones se maneja como si fuera una única instrucción se habrá creado un procedimiento. Este, a su vez, puede constar de otros bloques independientes. Por otra parte



Los bloques de repetición pueden ser de dos tipos. Cuando la comprobación de la variable de condición se efectúa antes de ejecutar el bloque de programa correspondiente, se llama bloque While (Mientras).

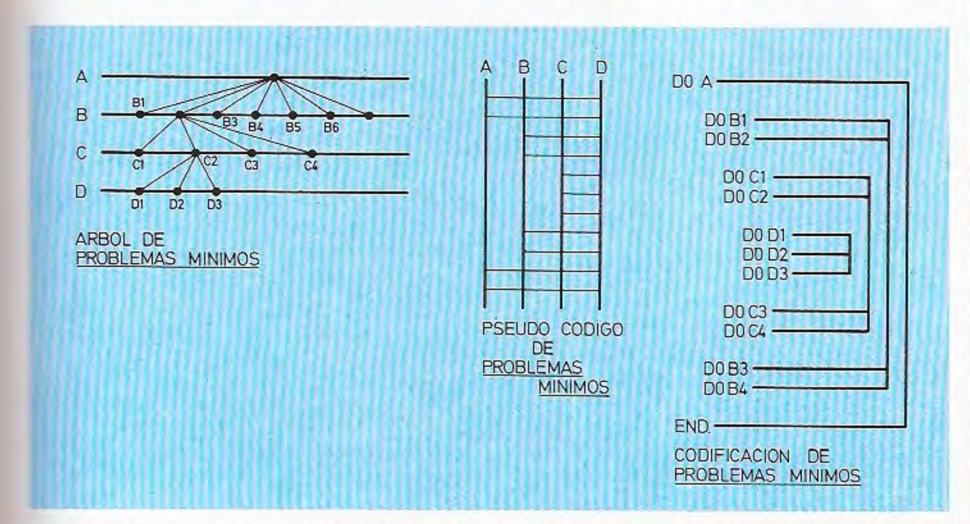
cada bloque de instrucciones utiliza variables que no deben ser siempre las mismas. Las variables de trabajo de un bloque son transferidas desde el bloque superior a través de los «argumentos» de entrada al procedimiento. Los resultados obtenidos pueden, a su vez, emplearse en otros procedimientos.

Las variables implicadas en un procedimiento pueden ser de uso exclusivo de este bloque o compartirse con el módulo que realiza la llamada.

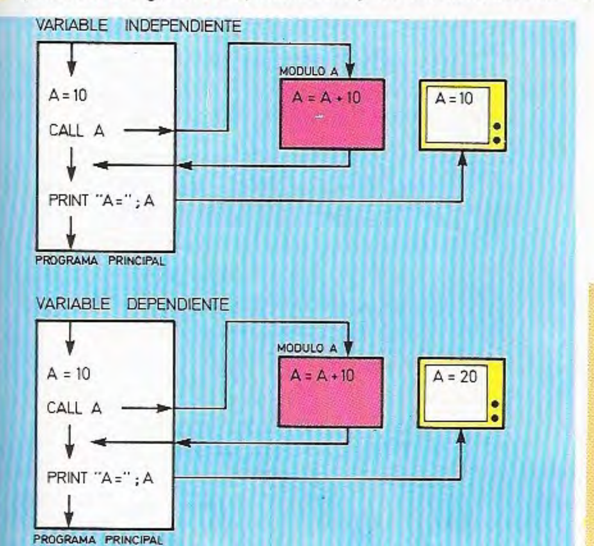
Para asegurar la flexibilidad de estos módulos, cada una de las partes de que se compone debe ser autónoma y contener, por lo tanto, sus propias variables independientes. Las modificaciones sucesivas no presentarán de esta forma problemas, ya que cada variable se usa sólo en un módulo concreto.

Sentencias de bifurcación

Cuando un programa llega a una bifurcación decide, en función del valor que tome determinada variable, el procedimiento que debe ejecutar a continua-



Cualquier problema general se puede descomponer en un conjunto de problemas mínimos. La figura ilustra, esquemáticamente, la generación, la realización del pseudocódigo correspondiente y la codificación de éstos.



Las variables independientes creadas en el programa principal conservan su valor a lo largo de éste. Las variables dependientes, sin embargo, puede modificarse por operaciones efectuadas dentro de un módulo cualquiera.

Multiprogramación

Los ordenadores que controlan la ejecución simultánea de varios procesos requieren una adecuada gestión de los recursos, tanto de software como hardware.
Cuando el ordenador trabaja en multiproceso el problema que debe abordar es el de la asignación de tiempo a cada uno de los procesos en curso. Para ello cada uno de los procesos puede estar en uno de estos tres estados: bloqueado, en espera o en ejecución.
Un proceso está bloqueado cuando espera que ocurra algo para poder continuar su ejecución (normalmente que ocurra una entrada o una salida del sistema).
Un proceso está en espera cuando ya está listo para continuar su ejecución y sólo precisa que se le dé el control del procesador central.
Un proceso en ejecución puede pasar a cualquiera de los otros dos estados, mientras que al estado de ejecución sólo pueden pasar los procesos en espera.
La parte del sistema operativo que gestiona el paso de un estado a otro se llama «scheduler». Asigna para ello una prioridad a cada una de las tareas.
En ocasiones es el propio sistema quien confiere las prioridades, teniendo en cuenta para ello el tiempo que puede durar la ejecución de cada una de las tareas.
Este proceso de asignación de prioridades debe tender a equilibrar los recursos necesarios y disponibles, y evitar que muchas tareas estén bloqueadas mientras que la CPU está parada por no tener tareas en espera.

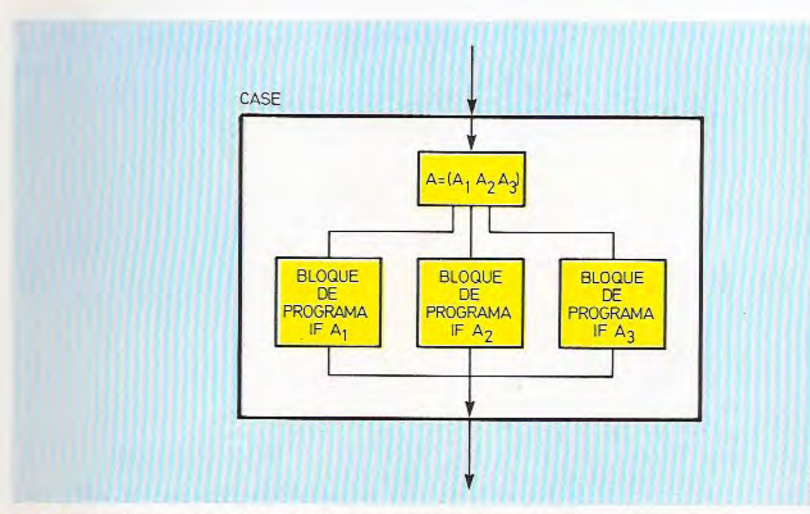
ción. La instrucción más común para realizar esta elección es CASE. El comando SELECT, a su vez, selecciona la variable de la que depende el procedimiento a llamar. Para cada uno de los valores —CASE— que tome esta variable se escoge un procedimiento u otro. Si sólo existen dos alternativas de elección posibles se usan sentencias del tipo «IF ... THEN ...» (si ... entonces...) o «IF ... THEN ... ELSE ...» (si ... entonces ... sino ...).

Las instrucciones de repetición pueden ser de varios tipos. En el primer caso el programa comienza por examinar la variable de condición y según el valor de ésta pasa a ejecutar el procedimiento repetitivo o no. En el segundo tipo la operación a repetir se ejecuta al menos una vez, se cumpla o no la condición impuesta, pues la comprobación de la variable de condición se efectúa inmediatamente después de la operación.

Utilizando este tipo de estructuras la tarea del programador se reduce a aplicarlas a problemas reales. Puede comenzar su labor traduciendo los algoritmos a módulos de pseudocódigo que se descomponen en procedimientos de nivel cada vez más bajo.

Descendiendo a estos niveles cualquier programador ajeno a la escritura del programa será capaz de leer, entender y modificar el programa.

Un bloque de repetición es del tipo Repeat (Repetir), si finaliza por la comprobación de la variable de condición.



Las estructuras de tipo Case proporcionan una gran potencia de programación a los lenguajes estructurados. El programa ejecuta un conjunto de instrucciones diferentes para cada valor de la variable de condición A.

Para saber más

¿Cómo se ejecutan varios programas que estén residentes en memoria al mismo tiempo?

En los sistemas pequeños el control de las interrupciones del sistema permite saltar de un programa a la posición donde está otro de nuestros programas. En equipos mayores, el sistema operativo se encarga de la tarea de distribución del tiempo y del espacio para cada uno de los posibles módulos que en un momento dado residen en la memoria.

¿Es posible hacer un programa con módulos escritos en distintos lenguajes como ASSEMBLER, COBOL y BASIC?

Es posible y, dependiendo de la máquina, puede ser más o menos sencilla la integración y el uso de estos elementos. De cara a la programación se puede ver a un conjunto de módulos que ejecutan un proceso determinado, bien como un todo uniforme y compacto, bien como un conjunto de tareas que bifurcan a direcciones absolutas donde está almacenado cada módulo, a través de sentencias de carga de programa objeto.

¿Es directamente ejecutable el pseudocódigo?

No. El pseudocódigo es un medio de representar la estructura interna de un programa. Tiene la ventaja de acercarse bastante al código final y de ser, además, fácil de leer y escribir.

¿Es suficiente que un programa no contenga instrucciones GOTO para que sea estructurado?

No es suficiente. La carencia de instrucciones GOTO es una característica muy importante de los programas estructurados, pero no es la única condición requerida para que lo sean. Otros factores, como el nivel de secciones independientes de que conste la codificación, son más representativos de este tipo de técnicas.

Ficheros

Tipos, organización y técnicas de acceso

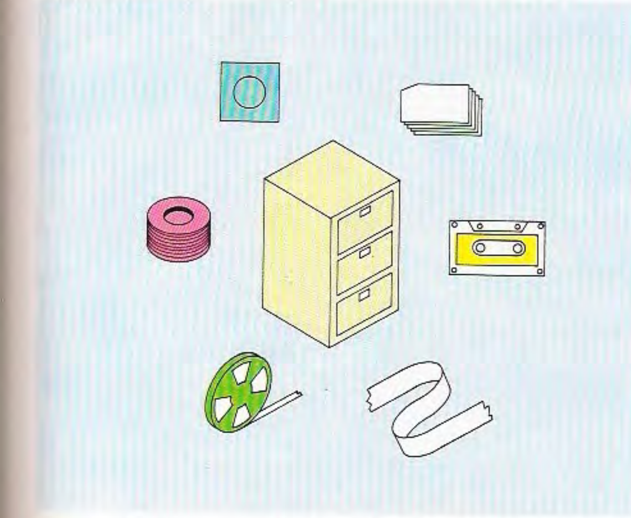


Para realizar cualquier tarea de proceso de datos se necesitan archivos que contengan la información a tratar. En los procesos de informática de gestión son tan importantes los archivos como los propios cálculos, que suelen ser muy sencillos (en la mayoría de los casos se reducen a simples sumas, restas y algunas multiplicaciones).

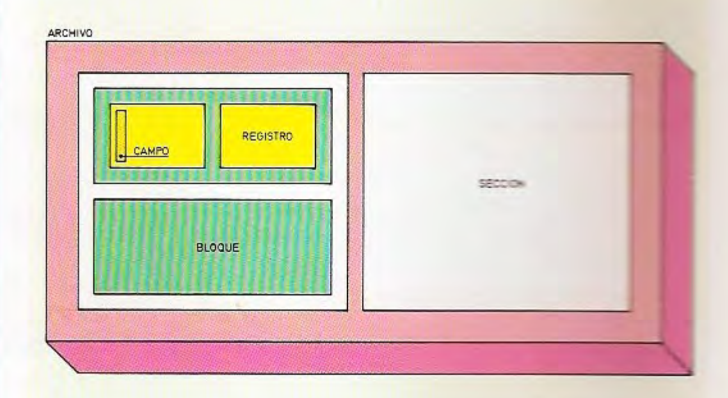
Se podría definir un archivo como «un conjunto de datos almacenados y ordenados».

Cuando se visita una oficina puede observarse que se emplean unas «carpetas» con el rótulo de ARCHIVOS. En ellas se guardan todos los documentos relacionados con las diversas actividades de la empresa. Por ejemplo, existen archivos de facturas, de las nóminas de los empleados, cuentas por pagar, etc.

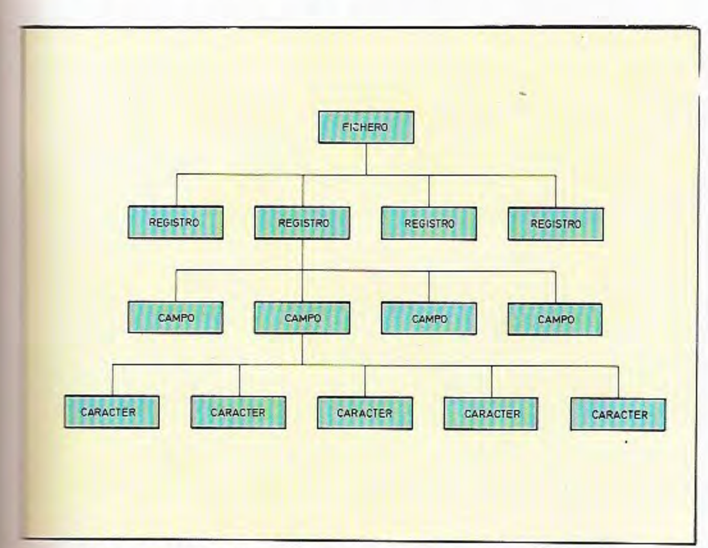
Esta forma clásica de guardar toda la información se revolucionó con la llegada de la informática. El tratamiento electrónico de la información ha hecho que en la actualidad, los archivos clásicos



La introducción de sistemas informáticos en la oficina moderna ha supuesto un cambio profundo en los sistemas de archivo. Esta transformación se cifra en que la información no se guarda ya sobre papel, sino en cinta magnética, discos o tarjetas.



El gráfico muestra las subdivisiones de un archivo electrónico: sección, bloque, registro y campo.



En la presente figura puede observarse el esquema elemental de un archivo de ordenador, subdividido según una estructura de «árbol».



El trabajo en una oficina mecanizada no hace necesario el empleo de papel como soporte principal de la información. Ahora es el teclado del ordenador el que hace las veces de bolígrafo y las unidades de almacenamiento magnético son las que sirven de soporte de la información escrita.

hayan sido sustituidos por un nuevo sistema en el que los ordenadores son los que organizan y tratan la información contenida en los archivos. En esta nueva organización, todos los archivos se suelen concentrar en un solo lugar denominado «biblioteca de archivos», situada cerca de la sala de ordenadores. Las carpetas de cartón se han sustituido por soportes legibles por el ordenador, en los que se almacena, de forma clasificada, la información.

La operación de almacenar los datos en estos medios de archivo recibe el nombre de *grabación* y la transferencia de esta información a la memoria interna del ordenador se denomina *lectura*. La lectura de un archivo no altera su contenido.

Para facilitar su tratamiento, los archivos de ordenador se subdividen en otros elementos. Aunque no todos los archivos tienen los mismos elementos, los más corrientes son:

- *Archivo*

Conjunto de datos ordenados. Un archivo está constituido por registros.

- *Sección*

Cuando un archivo es de gran tamaño se suele dividir en secciones. Cada una de estas secciones contiene un cierto número de bloques de registros. Las secciones pueden ser físicas o lógicas. No todos los archivos están divididos en secciones.

- *Bloque*

Los registros del archivo se agrupan en los llamados bloques de registros, que pueden contener desde uno hasta varios registros. El tamaño del bloque depende del medio disponible para almacenar el fichero, así como del tamaño de la memoria asignada a un bloque durante el proceso.

- *Registro*

Es un conjunto de datos que están relacionados y que se tratan como una unidad. Pueden ser de longitud fija o variable.

- *Campo*

Es una subdivisión de un registro y contiene datos numéricos, alfabéticos o alfanuméricos.

- *Carácter*

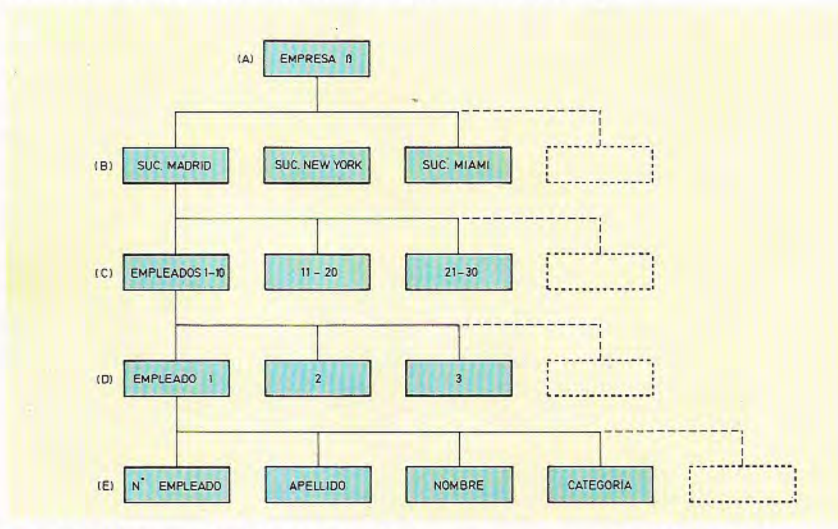
Es el elemento más pequeño del archivo.

Tipos de archivos

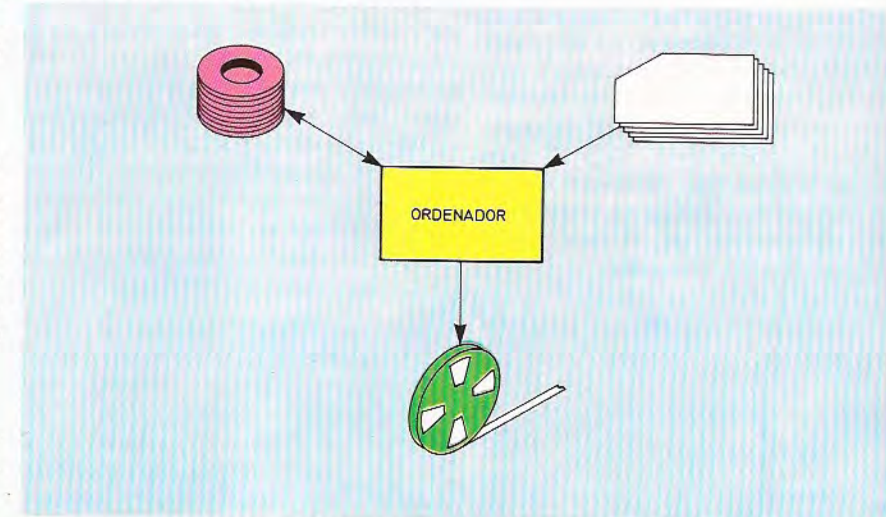
Los diferentes tipos de archivos se clasifican de acuerdo a su forma de utilización. Atendiendo a este criterio se dividen en:

– *Archivos de entrada*

Son aquellos que se utilizan para introducir información en la memoria del ordenador. También se denominan archivos de origen.



El gráfico muestra el posible esquema de organización interna de un archivo (A) en el que está registrada la información sobre el personal de una empresa. Las secciones (B) incluyen la información de cada sucursal, y en cada registro (D) se almacenan los datos referentes a cada empleado (campos-E).



El soporte de información en cinta magnética suele emplearse para archivos con una organización de tipo secuencial. Los discos magnéticos se emplean para ficheros organizados de forma directa o indexada. Las tarjetas perforadas son apropiadas para archivos de entrada.

– *Archivos de salida*

Se utilizan para almacenar información extraída de la memoria interna del ordenador; también se denominan archivos de destino.

– *Archivos de entrada/salida*

Se emplean tanto como archivos de origen como en funciones de archivos de destino de la información procesada por el ordenador.

Cuando se cambian los datos de un archivo para reflejar en él nuevas informaciones se dice que se está «actualizando».

Un ejemplo típico de archivo de entrada/salida es el utilizado para la expedición de un billete de avión. El operador de la oficina de viajes llama desde su terminal al archivo donde se encuentran las plazas disponibles de cada vuelo (archivo de entrada), lo actualiza indicándole que una de las plazas ha sido ocupada (archivo de salida) y, de nuevo, el archivo queda dispuesto para la próxima consulta.

Organización de los archivos

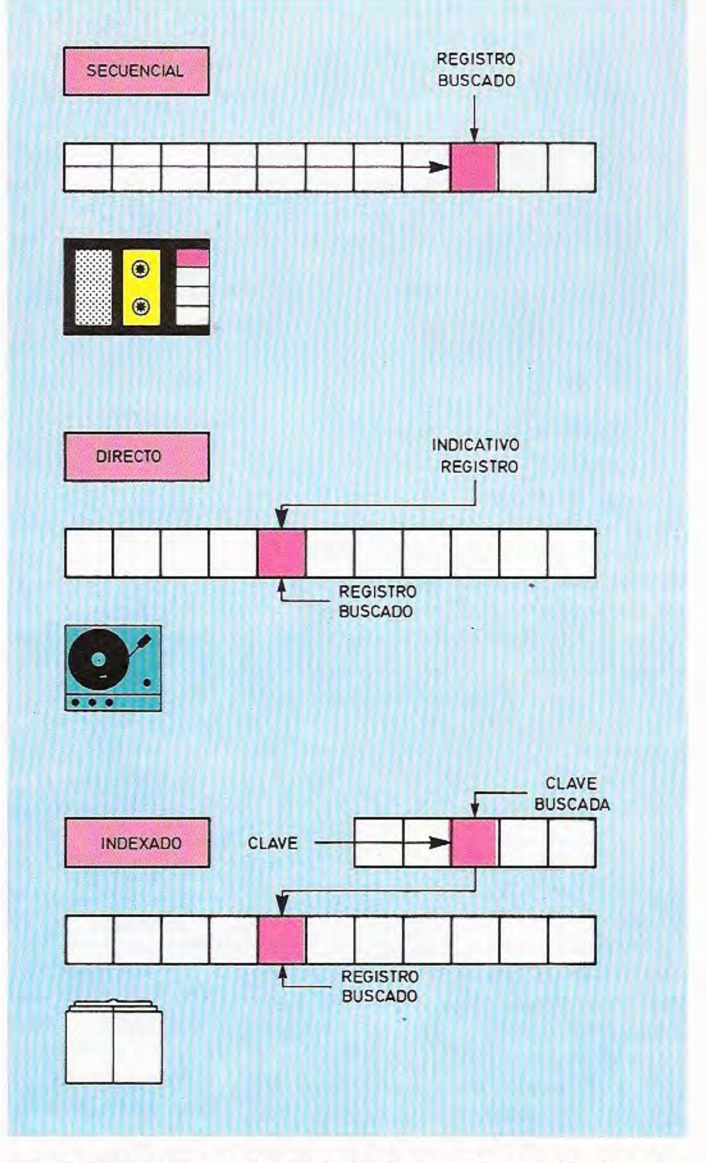
La diversa naturaleza de la información a almacenar se traduce en la exis-



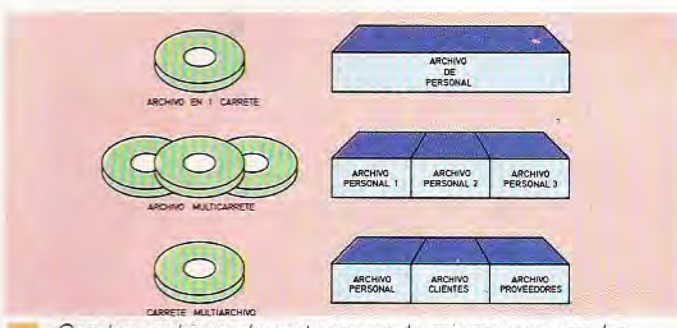
La actualización de la información contenida en ficheros electrónicos puede ser realizada en la actualidad por personal no especializado, debido a la carrera de simplificación que los fabricantes de software sostienen desde hace algunos años.



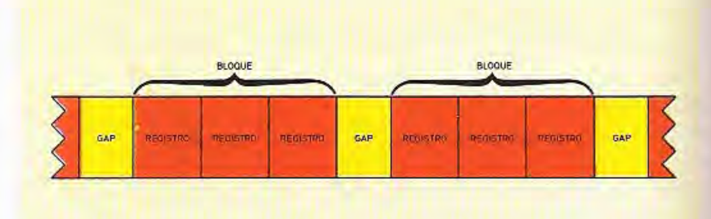
Un oficina puede definirse como un centro donde se recibe, manipula y genera información. En la actualidad, los sistemas de procesos de datos tienden, cada vez con mayor intensidad, a utilizar soportes de tipo magnético: discos, cintas, casetes, etc., reservando el papel para salidas de información ya elaboradas.



El gráfico muestra las distintas formas de organización de un archivo de ordenador: secuencial —para buscar un dato es necesario recorrer todo el fichero—; directo —el acceso a la información se realiza como en un tocadiscos: directamente—; indexado —que permite la búsqueda de un dato como si de un listín telefónico se tratara.



Con los archivos de cinta ocurre lo mismo que con los archivadores clásicos. Pueden almacenarse varios archivos en un mismo archivador o, por el contrario, pueden ser necesarios varios archivadores físicos para almacenar un solo archivo.



Los GAP son zonas en blanco de la cinta que sirven para separar los bloques de información. Estos «huecos» permiten a la unidad de cinta alcanzar la velocidad de grabación o lectura adecuada entre cada arranque y parada.

tencia de archivos con distinta organización. Por ejemplo, aun aplicando los métodos clásicos, no se archivan de la misma forma las facturas y la correspondencia.

Esta diversidad en los métodos de almacenar la información da lugar a tres técnicas básicas de organización de los archivos.

- *Organización secuencial*

En ella los registros están grabados unos a continuación de otros. Hay que leerlos o actualizarlos en el mismo orden en el que están grabados. La información registrada en cinta magnética pertenece a este tipo de organización. Tiene el problema de que para acceder a cualquier registro hay que pasar por todos los registros anteriores, con lo que resultan muy lentos. Es útil cuando se quiere almacenar una información que debe ser leída de forma completa y ordenada.

- *Organización directa*

En ésta se puede acceder a una determinada información directamente, sin necesidad de pasar por las informaciones grabadas previamente. Para conseguirlo el programador crea unas claves indicativas de cada registro, relacionadas con la posición en la que están grabados. El medio de soporte para este tipo de organización suele ser el disco magnético. Un ejemplo de esta modalidad de organización sería un archivo de cuentas corrientes.

- *Organización indexada*

Los registros se graban de forma secuencial; si bien, se crean unas tablas o índices que permiten el acceso directo a cualquier tipo de información. El medio de almacenamiento utilizado con esta técnica de organización es, asimismo, el disco magnético. El sistema es análogo al índice alfabético de un libro. La búsqueda en el índice no es secuencial, ya que está ordenado alfabética o numéricamente. Un ejemplo sería un fichero de información bibliográfica.

Medios de archivo

Como ya se ha visto, los archivos de un ordenador deben residir en un medio que pueda ser leído o grabado por éste. En un principio se utilizaron como medios de almacenamiento de información tanto las tarjetas perforadas y las cintas perforadas de papel, como las cintas magnéticas, discos y tambores magnéticos. En la actualidad se han impuesto los medios magnéticos, debido a que presentan una serie de ventajas sobre los otros dispositivos. Estas ventajas se pueden resumir en las siguientes:

– Los medios magnéticos poseen una velocidad de transferencia de datos que es ideal para el tratamiento más eficaz de la información en archivos. Las tarjetas perforadas y las cintas de papel son demasiado lentas.

– Los medios magnéticos poseen una duración mucho mayor que la de los medios perforados.

– Los medios magnéticos pueden borrarse para reutilizarlos con una nueva información.

– A igualdad de espacio físico, los medios magnéticos almacenan más información que los medios perforados y permiten una longitud de registro prácticamente ilimitada.

– Los medios magnéticos son más baratos por carácter almacenado que los medios perforados, debido a su gran capacidad.

Archivos de cinta

Los registros se almacenan en orden secuencial y se procesan en serie. Es decir, el registro primero se procesa antes que el segundo, el segundo antes que el tercero, etc. Si se quisiera procesar los registros en un orden distinto, el operador tendría que recorrer la cinta completa para ir localizando cada registro, lo que acarrea una notable pérdida de tiempo.

Los registros del archivo en cinta magnética se agrupan en bloques separados por trozos de cinta sin grabar (GAP). La necesidad de esta agrupación de los registros es debida a que para leer la cinta, la unidad debe acelerar desde una velocidad cuando cero hasta la velocidad de lectura y, a continuación, disminuir la velocidad cuando se haya completado la lectura. Si se leyera de registro en registro estos tiempos de parada/aceleración serían interminables. Lo mejor es leer o grabar ***bloques*** para ahorrar tiempo y espacio de cinta. El tamaño de los bloques viene asignado por el programa.

Se llama *cabecera* a la parte de cinta en blanco situada al comienzo del carre-

te y *cola* a la parte de cinta en blanco al final.

Para que el operador sepa cómo colocar correctamente el carrete, existen dos marcas especiales al comienzo y al final de la parte útil de la cinta. La primera, llamada «BOT» (Begining Of Tape, principio de cinta) y la segunda, «EOT» (End Of Tape, final de cinta).

Con objeto de que los usuarios de los ficheros puedan identificar los carretes éstos tienen una etiqueta exterior donde aparece el nombre y el número del archivo. Suelen llevar también una etiqueta grabada magnéticamente con el mismo fin que la anterior.

Puede ocurrir que un archivo ocupe un solo carrete o que ocupe, debido a su tamaño, más de un carrete, así como que un carrete almacene varios archivos. Veamos cómo se dispone la información en estos tres casos.

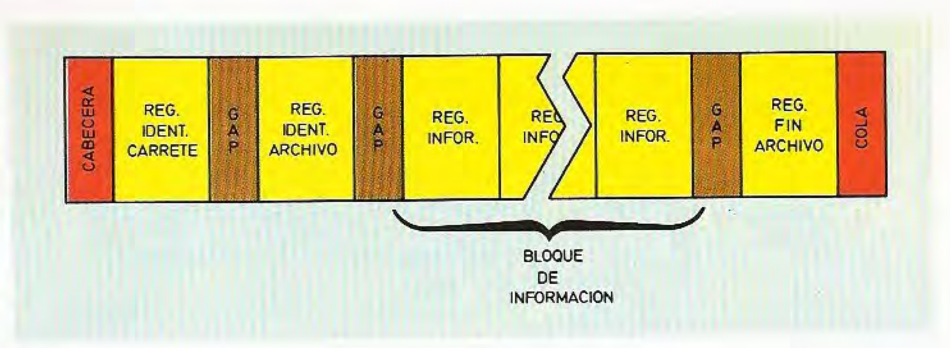
- En el archivo de un solo carrete existen dos bloques de un registro, situados al comienzo de la parte útil y separados uno de otro por un GAP, que indican la identificación del carrete y la del archivo respectivamente.

A continuación está el primer bloque de registros de información o datos. A este bloque le seguirán los restantes del archivo. El final del archivo lo indicará un bloque de un solo registro, llamado de control.

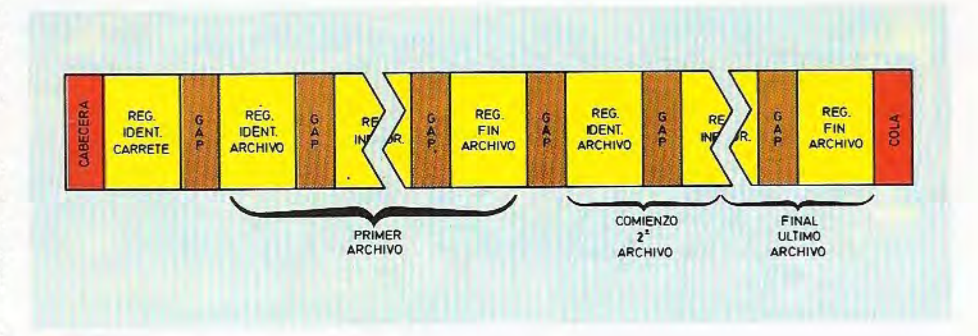
- Cuando un archivo ocupa varios carretes –multicarrete–, la parte del mismo que ocupa cada carrete debe ser indicada en la etiqueta exterior de cada uno de ellos. El final del primer carrete contiene solamente un registro que indica el final de cinta. En el siguiente carrete hay una etiqueta que ocupa dos bloques: el primero, que indica el registro de identificación del carrete, y el segundo, que indica el registro de identificación de datos del archivo, precisando que estamos en el segundo carrete. El último bloque del carrete número 1 y el primero del número 2 deben ser bloques completos, no partes de un bloque.

La parte de un archivo multicarrete contenida en un carrete es a lo que se llama *sección* física del archivo. Si un archivo ocupa dos carretes se dice que contiene dos secciones físicas. El programa que procese este tipo de archivos debe contener las instrucciones necesarias para cambiar automáticamente de una sección a otra. Si se dispone de las unidades necesarias para hacerlo, el programa debe darle tiempo al operador para montar la cinta.

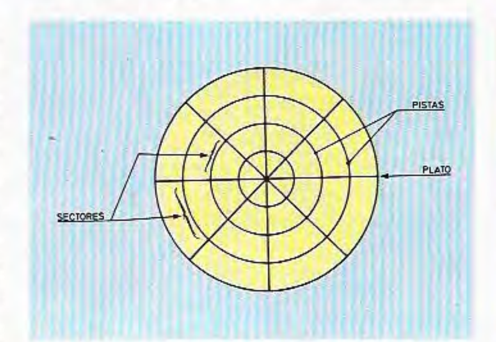
- Si un carrete tiene más de un archivo, se colocan marcas de control y etiquetas que identifiquen cada archivo. Estos carretes reciben el nombre de *multiarchivo.* Para localizar un archivo determinado se debe buscar desde el principio de la cinta el registro de identificación.



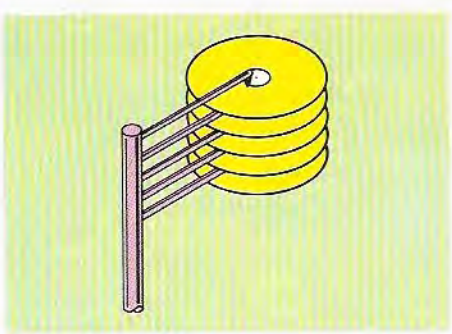
Organización de un archivo en un solo carrete de cinta magnética.



En la figura puede observarse la organización de un carrete de cinta en la que se encuentran grabados varios archivos (organización multiarchivo).



Organización de la información en discos magnéticos: pista —zona del disco que recorre el cabezal en una vuelta completa— y sector —divisiones dentro de cada pista.



Disposición en cilindro de una unidad de almacenamiento en disco rígido.

Archivos públicos, privados y compartidos

En la actualidad, la mayor parte de las empresas importantes de cualquier sector utilizan el almacenamiento electrónico de datos para conformar sus archivos. Estos contienen la información referente a la actividad de la empresa. A esos archivos se puede acceder directamente, en el lugar en el que se encuentra el centro de cálculo, o bien a través de terminales que pueden estar alejados geográficamente del centro.

En el proceso de expedición de un billete de avión no conocemos dónde se encuentra el centro de control de billetes; no obstante, desde cualquier terminal el operador puede reservar un billete y actualizar el fichero que contiene la información sobre los diferentes vuelos. Este es un ejemplo de fichero *Público,* que puede ser modificado por cualquier usuario sin que nadie lo impida. Podría definirse el Fichero Público como todo archivo al que puede acceder el usuario para leerlo y modificarlo cuando sea necesario. El único impedimento para acceder a este tipo de fichero es la clave indicativa que se le asigna a cada usuario, pero cualquier persona que la conozca puede manipular este tipo de ficheros.

Existen también informaciones referentes a actividades que por diversos motivos no es aconsejable que sean conocidas, bien por los usuarios de los terminales o bien por algunas personas del centro informático de la empresa. Esas informaciones están también almacenadas en ficheros, aunque en unos ficheros especiales a los que se cataloga como *Privados.* Estos ficheros se protegen contra posibles accesos no autorizados. Una forma de protección es utilizar las llamadas «Claves ensambladas» que actúan como un sistema de alarma; forman parte del hardware del equipo y hay que conocerlas para poder desactivarlas.

Por último, pueden existir ficheros en los cuales haya informaciones de los dos tipos: públicas y privadas. Este tipo de ficheros son los *compartidos.* Un ejemplo de éstos pueden ser los de una entidad bancaria. Cuando el cuentacorrentista desea conocer su saldo o sacar dinero, el operador del terminal teclea su clave y actúa sobre el archivo que contiene las cuentas corrientes. Sin embargo, desde ese terminal también se pueden obtener otras informaciones confidenciales sobre la actividad bancaria del cliente, que sólo competen, por ejemplo, al director.

Archivos en casete

Vamos a terminar el estudio de la cinta magnética hablando de otro soporte magnético, muy utilizado actualmente en el mundo de los ordenadores domésticos; se trata del casete de cinta magnética.

Se emplea para almacenar archivos de escaso tamaño. Tiene dos pistas para grabar y una etiqueta que identifica la pista a la que se accede.

El área de grabación es más estrecha que la de la cinta en carrete y la densidad de grabación también es menor. La organización de los archivos es, en líneas generales, la misma que en la cinta magnética en carrete para minis y grandes equipos.

Archivos en disco

Así como la cinta magnética es un medio ideal para grabar los registros de un archivo en orden secuencial, el disco magnético es el medio utilizado con mayor eficacia para leer y grabar registros

TIPOS DE PERIFERICOS Y ARCHIVOS

TIPOS DE MEDIOS O PERIFERICOS	TIPOS DE ARCHIVOS: Entrada	Salida	Entrada/Salida
Soporte de papel (tarjetas, cinta perforada MICR, OCR)	Entrada original (Proceso en batch)	Archivos maestros actualizados, archivos de respaldo (Proceso en batch)	No aplicable
Medios magnéticos de acceso secuencial (cinta magnética casete)	Entrada original Archivos transacciones y archivos maestros utlizados para actualización (Proceso en batch)	Archivos maestros actualizados archivos de respaldo (Proceso en batch)	Uso no práctico por la dificultad de regrabación en el mismo sitio
Medios magnéticos de acceso directo (Disco, tambor)	Archivos maestros y de transacciones utilizados para actualización (Proceso en batch)	Archivos maestros actualizados (Proceso en batch)	Archivos maestros actualizados (Proceso en batch y en tiempo real)
Tarjeta de banda magnética	Entrada para obtención de informes o análisis de contenido	Cuando se crea inicialmente el archivo maestro	Archivos maestros actualizados (Proceso en batch)
Impresora	No aplicable	Informes y listados de respaldo	No aplicable
Terminales	Entrada de transacciones (Proceso en tiempo real o remoto en batch)	Respuestas en procesos de tiempo real y en línea	No aplicable

a los que se quiere acceder directamente.

La lectura y grabación en las superficies del disco se realiza por medio de las llamadas cabezas de lectura y grabación.

Un archivo de disco se organiza a partir de varias unidades que se enumeran a continuación:

- Carácter: representado normalmente por 8 bits de información (1 byte).
- Sector: contiene generalmente 512 caracteres.
- Pista: se llama así a la superficie del disco recorrida por la cabeza durante un giro completo del disco. Cada pista contiene normalmente 8 sectores.
- Disco: contiene un cierto número de pistas. Existen unidades en las que el disco contiene hasta 1.024 pistas. Cada pista contiene el mismo número de informaciones, por lo que las pistas centrales, de menor longitud que las periféricas, se graban con mayor densidad.

La longitud de los «bloques» del disco viene determinada por el ordenador, y pueden ocupar de 1 a 8 sectores.

Un archivo en disco también se pude dividir en secciones que pueden estar formadas por varios sectores contiguos del disco. Un archivo puede ocupar dos o más secciones de un disco o se pueden tener varias secciones de un archivo en discos diferentes.

Cuando varios archivos o secciones de archivos se encuentran en un solo disco, es necesario crear el llamado «Directorio de archivos de disco», que contendrá la información referente al nombre del archivo, su situación dentro del disco, fecha de grabación, etc.

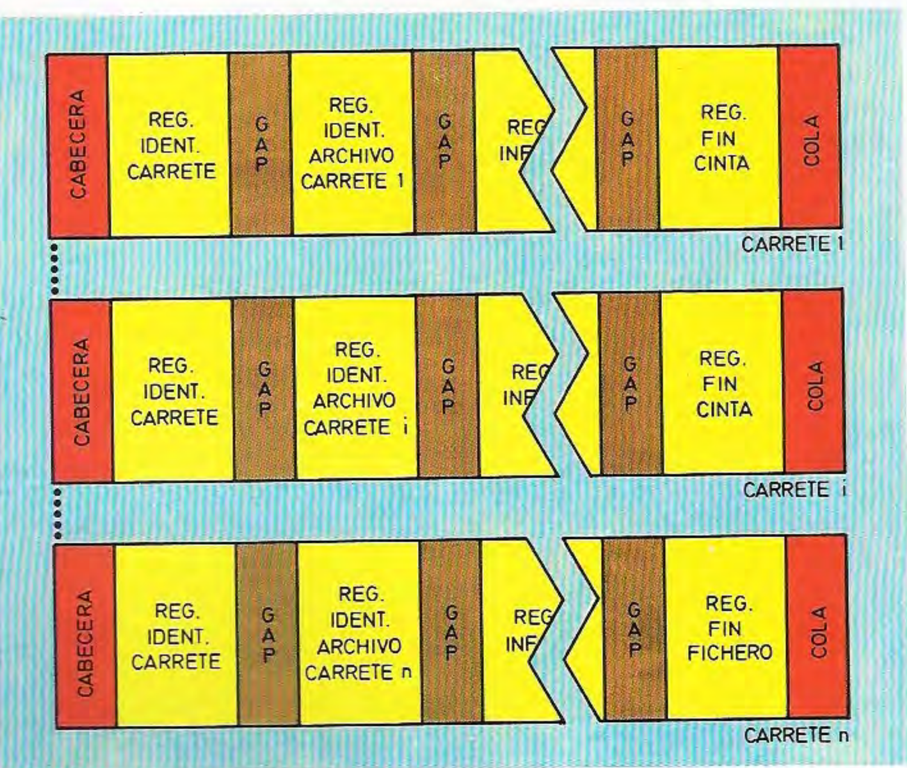
El lector puede tener problemas a la hora de distinguir entre un sector y una sección en un archivo de discos. Para evitar confusiones siempre hay que recordar que el sector es una división «física» del disco, mientras que la sección es una división «lógica» del archivo, realizada por el programador.

Evidentemente, existe una estrecha relación entre las unidades físicas de un archivo (disco, pista, sector) y las unidades lógicas (sección, bloque). Por ejemplo, un bloque está contenido, normalmente, en un sector.

Una organización clásica de archivos en disco es el llamado «cilindro». En un



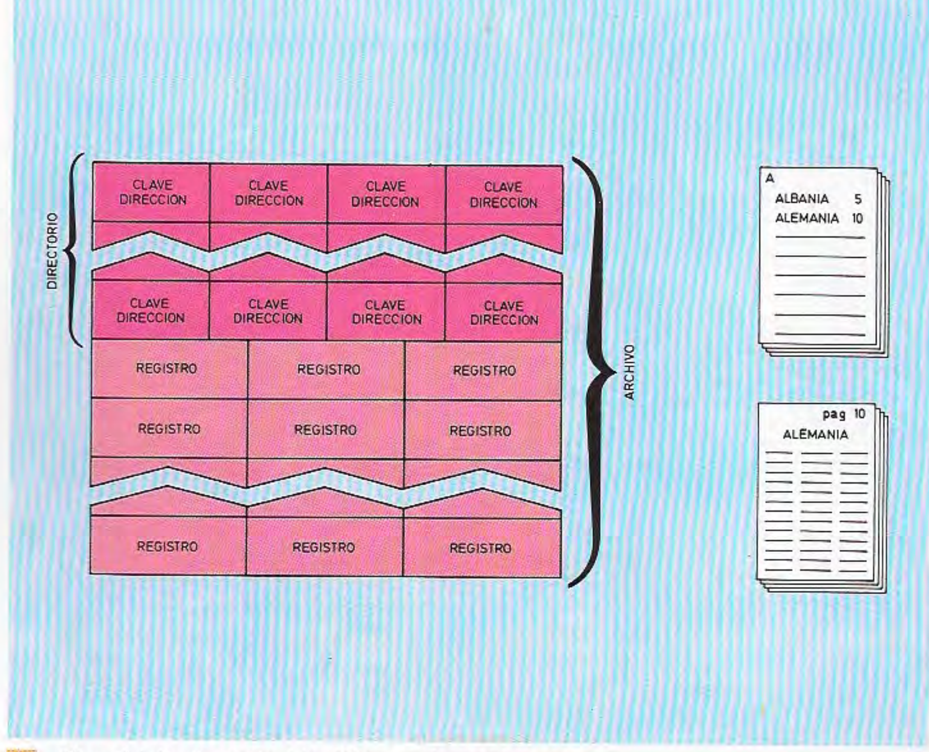
Los soportes magnéticos se han impuesto sobre el papel y las tarjetas perforadas como medios de archivo, debido a su mayor economía —relación precio/calidad— y a la posibilidad de reutilizarlos innumerables veces.



Cuando un archivo es muy extenso, son necesarios varios carretes de cinta para su completo almacenamiento. El gráfico muestra la organización de un archivo multicarrete; el último bloque de cada carrete debe ser un bloque completo.

Directorio de archivos

En un paquete de discos pueden existir almacenados varios archivos o varias secciones de un archivo. Para poder identificarlo completamente se usa lo que se llama un «directorio de archivo».
El directorio contiene el nombre que se le asigna al archivo, la posición del archivo o de su sección en el disco, la fecha de grabación o actualización y también la fecha en que caduca la información almacenada.
El directorio se crea cuando se va a grabar la información en el disco. Es el sistema operativo del ordenador el que realmente controla el directorio grabándolo y manteniéndolo. Cuando el usuario quiere saber de qué archivos está compuesto un paquete de discos, tiene que acceder al directorio con la ayuda del sistema operativo del ordenador. De lo dicho se deduce inmediatamente que si se modifica un archivo, es decir, se actualiza, esa modificación debe aparecer en el directorio del archivo y es el sistema operativo el encargado de reflejarlo.
Cuando un paquete de discos contiene varios archivos, cada uno de éstos tiene una entrada en el directorio que le es asignada por el sistema operativo cuando se graba. Si el usuario quiere saber si puede disponer de un determinado archivo para procesarlo, tiene que acceder al directorio, con el objeto de cerciorarse de su disponibilidad.
En todos los sistemas existe una rutina de utilidad que permite a cualquier programador imprimir el directorio y estudiar la configuración del paquete de discos.
Para ayudar al programador a localizar el sector que contiene un registro determinado se pueden crear directorios dentro del mismo archivo de datos.
Este tipo de directorios es típico de organizaciones de archivo directo e indexado. El directorio contiene la clave del registro y la dirección y es en realidad un archivo de referencia creado por el sistema operativo o por el mismo programador. A este tipo de directorio se le conoce con el nombre de directorio de índice de claves.



Con objeto de poder identificar cada archivo y sus correspondientes registros se crea el denominado «directorio del archivo». El directorio guarda un gran paralelismo con el índice alfabético de un libro.

paquete de discos con cabeza de lectura y grabación para cada cara, el archivo se dispone de la siguiente manera: se comienza a grabar el archivo en la primera pista del primer disco; cuando se graba esta pista, el archivo no continúa en la segunda pista del primer disco, sino en la primera del segundo disco; se completa ésta y se pasa a la primera del tercer disco y así sucesivamente. Con esto se logra que la lectura del archivo se haga rápidamente sin tener que esperar a que existan movimientos de las cabezas, ya que mientras se lee una pista de una cabeza la siguiente ya está preparada para continuar la lectura. Se asemeja a un cilindro formado por todos los platos y de ahí le viene su nombre.

Archivos en disquete

El disco magnético tiene un «hermano menor», llamado disquete, disco flexible o floppy. Tiene menor capacidad de almacenamiento y se utiliza como elemento de almacenamiento de programas y datos. Lleva una etiqueta exterior para su identificación y su organización interna es similar a la del disco rígido descrita en el epígrafe anterior.

Tambor magnético

Vamos a terminar haciendo una breve descripción del tambor magnético, dispositivo que en la actualidad se utiliza escasamente debido a que tiene muchos problemas, en comparación con la cinta y el disco, a la hora de intercambiarlo.

La información se graba en pistas, cada una de las cuales tiene su propia cabeza. Se utiliza de una manera más eficaz que la cinta cuando se quiere acceder a una parte determinada del archivo y es un medio ideal para almacenar archivos permanentes. Utilizan directorios y los archivos en tambores se pueden dividir en *segmentos* a los que se puede acceder individualmente, por lo que se pueden utilizar como medios de acceso directo.

Acceso a archivos

En el trabajo diario de un centro de proceso de datos, o en una oficina dotada de sistemas informáticos, resulta imprescindible buscar la información almacenada en archivos. El conjunto de técnicas cuyo objeto es facilitar la búsqueda de datos en archivos electrónicos recibe la denominación de «acceso a archivos».

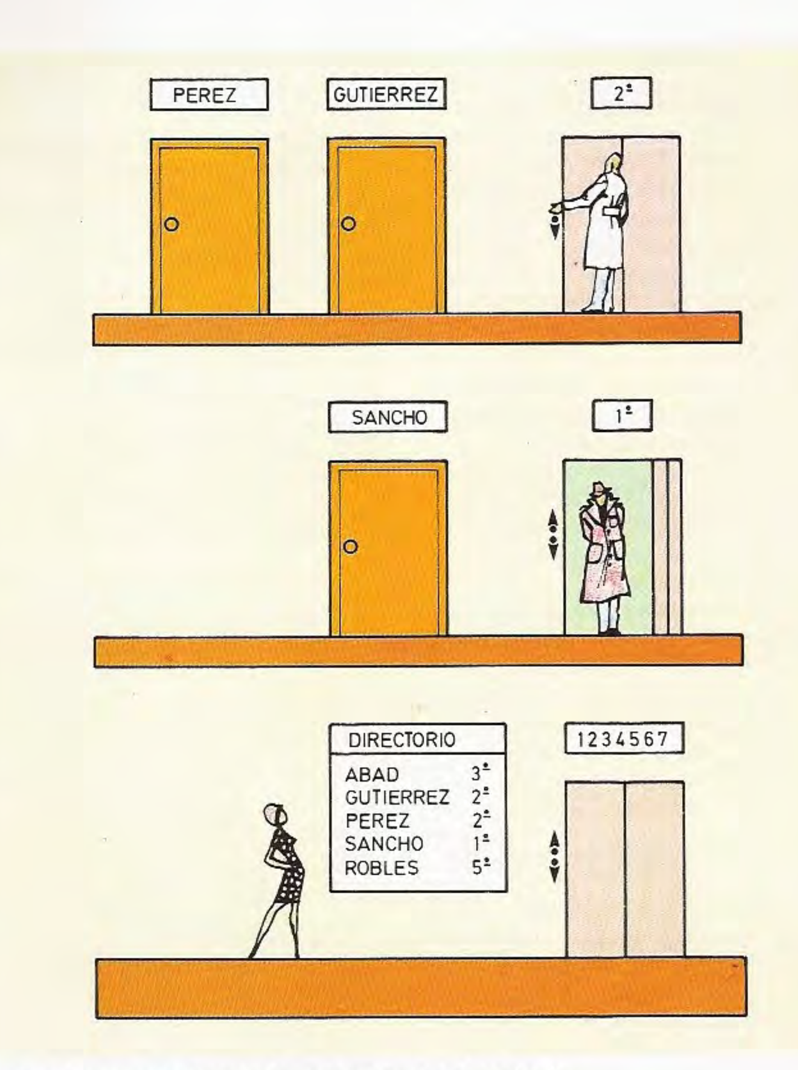
En un capítulo anterior hemos analizado la forma en la que se organizan los archivos de acuerdo a tres técnicas: secuencial, directa e indexada. Por supuesto, a cada una de estas técnicas de organización corresponde un método de acceso específico.

El sistema operativo del ordenador es el encargado de organizar la sucesión de operaciones necesarias para el acceso a los archivos; para ello, cuenta con un conjunto de rutinas o programas especializados en el acceso a archivos con una determinada organización.

Los cinco métodos de acceso más relevantes son los siguientes:

- Acceso secuencial.
- Acceso directo.
- Acceso indexado.
- Acceso particionado.
- Acceso virtual.

Los más utilizados son los tres primeros.

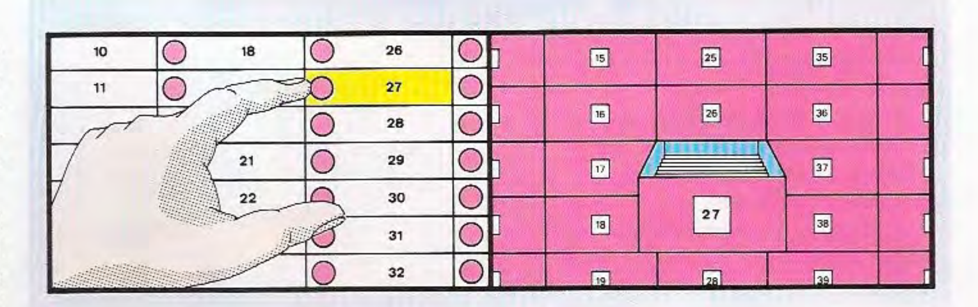


En el acceso particionado el ordenador encuentra la información pedida de la misma forma en la que una persona localizaría un despacho en un edificio de oficinas: buscando el piso en el directorio y localizando después la puerta de dicho piso.

Elección del método de acceso

Aunque no existen normas fijas para la elección del método de acceso, hay que tener en cuenta una serie de criterios, aplicables a la hora de definir un archivo, que condicionarán la forma adecuada de acceder al mismo.

En primer lugar se tendrá en cuenta el medio sobre el que se va a almacenar el fichero. Si por ejemplo, el medio es una cinta magnética o fichas perforadas, el método de acceso idóneo será el secuencial. Cuando el medio es un disco magnético se presentan varias posibilidades, debido a que es un dispositivo de acceso directo.



En el acceso virtual se carga la memoria con un número entero de bloques dentro de los cuales se busca el registro deseado. Si sólo dispusiéramos de ficheros manuales revisaríamos toda la información de una parte del archivo total.

Acceso Padre/Hijo

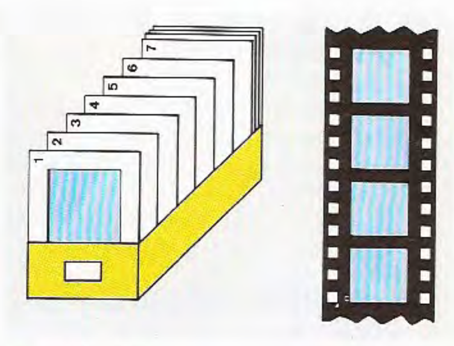
Este método es bastante usado como técnica de acceso secuencial. Intervienen tres tipos de archivos: el archivo maestro, el archivo de transacciones que va a actualizar el archivo maestro, y el nuevo archivo maestro. Al archivo maestro original se le llama «Padre» y al nuevo que se obtiene después de actualizarlo se le denomina «Hijo».

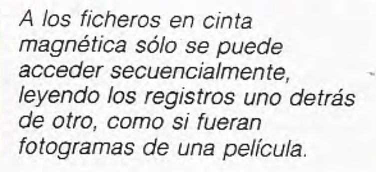
Es un método muy útil para actualizar archivos de gran actividad. En ellos es imprescindible que tanto el archivo maestro como el de transacciones estén clasificados en el mismo orden. Con este método se pueden insertar en el archivo «Hijo» nuevos registros, que deben llevar un tipo de código que denote que la transacción es una inserción.

Supongamos que se tiene un archivo maestro con registros cuyas claves son 1, 2, 3, 4, 5, 6. El archivo de transacciones tiene registros con claves 1, 2, 4, 5. Se supone que los números de registros del maestro y del de transacciones corresponden a códigos de productos de un almacén.

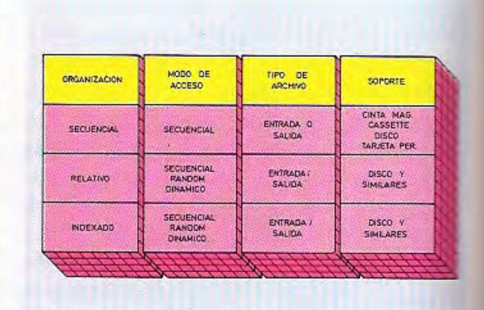
Este metodo sigue la siguiente lógica:

1. Se leen el primer registro de transacción y el primero del maestro. Si los números de producto son iguales (1 = 1), los datos del registro de transacciones actualizan al maestro. Una vez actualizado se manda grabar en el nuevo archivo maestro («Hijo»).
2. Se lee a continuación el siguiente registro de transacción y se compara con el siguiente del maestro; de nuevo se observa que son iguales (2 = 2), con lo que se procede como en el caso anterior.
3. Se lee a continuación el siguiente registro de transacción (4) y el correspondiente del maestro que es el 3. Esto indica que no hay que actuar sobre el maestro número 3 y que debe copiarse íntegramente en el nuevo archivo maestro. Se conserva el registro número 4 de transación y se sigue el proceso.
4. El registro de transacción número 4 se compara con el cuarto del maestro que es también el número 4. Se actualiza el correspondiente registro.
5. El proceso sigue de forma análoga hasta terminar. Puede ocurrir, por ejemplo, que se tenga un archivo maestro formado por los registros 1, 3, 4, 5 y 6 y el de transacciones por 1, 2, 4 y 6. El de transacciones incorpora un nuevo registro 2 que no lo tiene el maestro, lo cual puede evidenciar un error o indicar que se trata de una inserción.



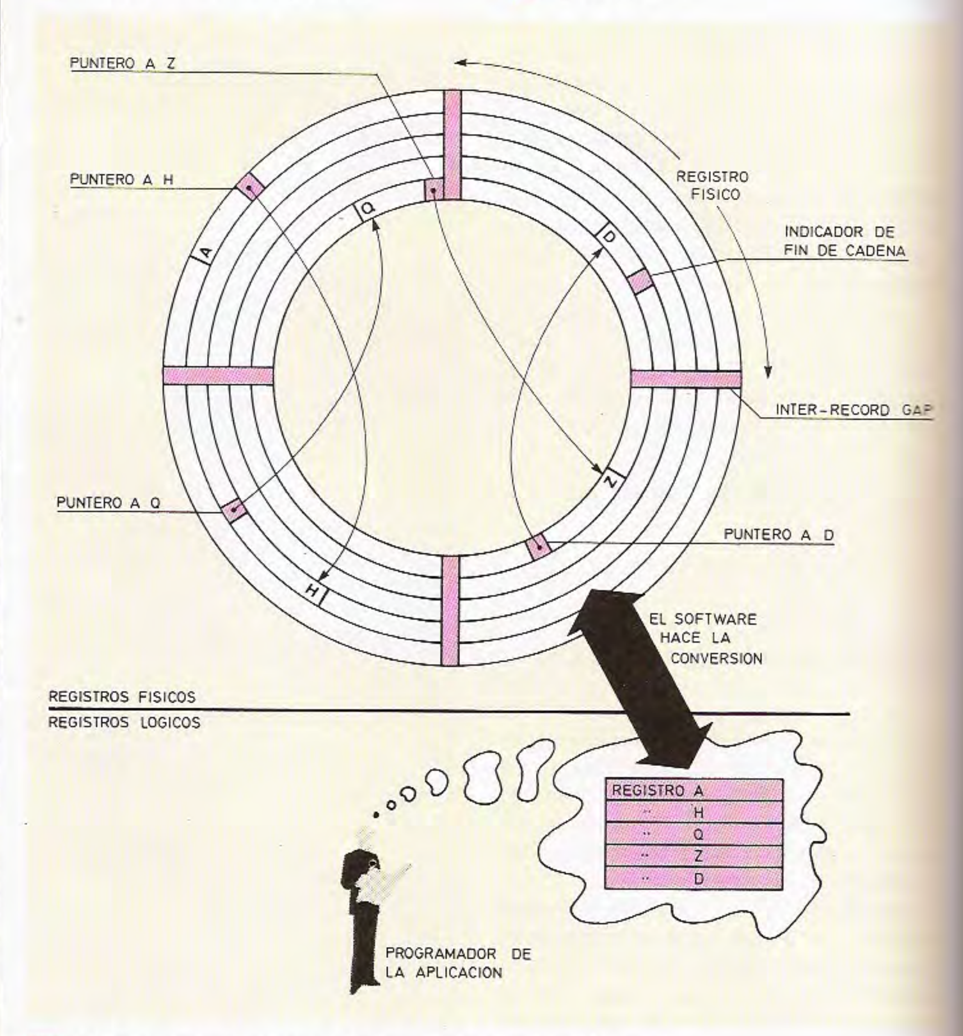


A los ficheros en cinta magnética sólo se puede acceder secuencialmente, leyendo los registros uno detrás de otro, como si fueran fotogramas de una película.



ORGANIZACIÓN	MODO DE ACCESO	TIPO DE ARCHIVO	SOPORTE
SECUENCIAL	SECUENCIAL	ENTRADA O SALIDA	CINTA MAG. CASSETTE DISCO TARJETA PER.
RELATIVO	SECUENCIAL RANDOM DINAMICO	ENTRADA / SALIDA	DISCO Y SIMILARES
INDEXADO	SECUENCIAL RANDOM DINAMICO	ENTRADA / SALIDA	DISCO Y SIMILARES

La organización, el tipo y el soporte físico de un fichero son los factores que determinan la forma de acceso más adecuada.



El almacenamiento real o físico de los datos dentro de la memoria no se corresponde con su organización lógica. El sistema operativo del ordenador se encarga de gestionar la memoria sin intervención directa del usuario.

En este caso hay que considerar factores tales como:

– *Actividad del fichero.* La actividad se refiere al número de registros accedidos durante un tiempo determinado. Para actividades bajas se utiliza mejor el acceso directo. Cuando la actividad es alta —mayor del 60 por 100; esto es, se actúa sobre el 60 por 100 de los registros del fichero— es preferible el acceso secuencial, secuencial indexado o virtual.

– *Volatilidad del fichero.* La volatilidad define la cantidad de registros que se pueden actualizar, borrar o añadir en un fichero ya existente, durante un tiempo determinado. Con baja volatilidad se puede acceder de forma secuencial indexada, mientras que con alta volatilidad resulta más oportuno el acceso particionado.

– *Tiempo de respuesta a las consultas.* Hoy día se requiere en muchas aplicaciones un tiempo de respuesta muy bajo, por lo que es conveniente ir a organizaciones y métodos de acceso directo.

Acceso secuencial

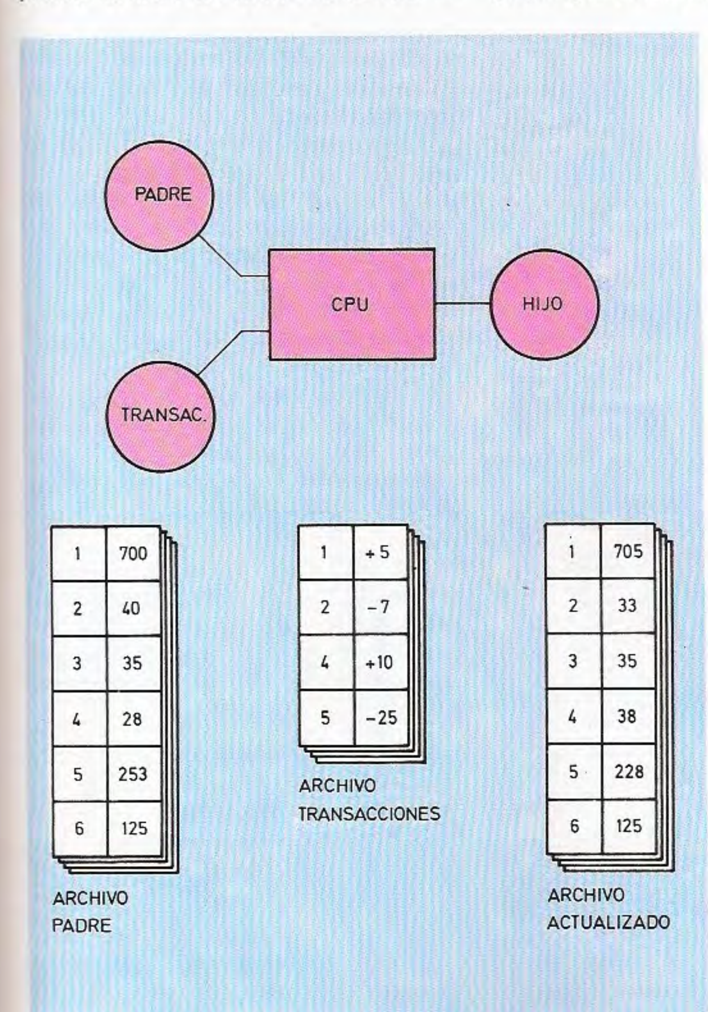
En el acceso secuencial se llega a los registros en el mismo orden o secuencia en que están organizados. Por ejemplo, si queremos llegar al registro 24 hay que leer primero los 23 que le anteceden.

Mediante el método secuencial se accede a archivos organizados de forma secuencial.

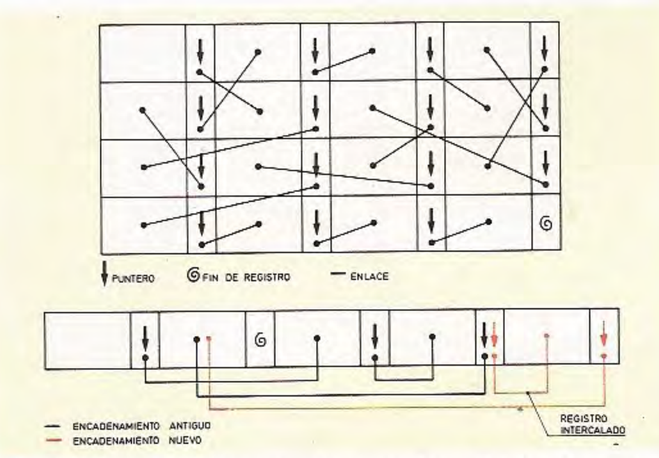
Existen tres técnicas de acceso secuencial muy utilizadas: la de análisis de contenido, la padre/hijo y la secuencial-selectivo.

Análisis de contenido

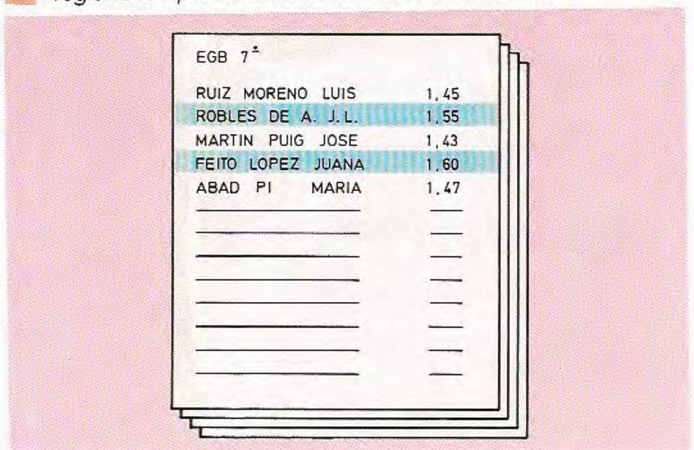
Se utiliza par acceder y procesar un archivo en el que hay que examinar cada



En el método Padre/Hijo, el archivo de transacciones modifica parte de la información contenida en el archivo Padre, que una vez actualizada se almacena en el archivo Hijo.



Los punteros se encargan de encadenar los registros de un fichero, con independencia de la posición física que ocupen éstos dentro de la memoria. Cuando se intercala un nuevo registro los punteros involucrados se modifican.



La selección de todos los alumnos de un curso que midan más de un metro cincuenta no tiene por qué hacerse siguiendo el orden alfabético. De igual forma, el acceso por análisis selectivo no sigue un orden lógico.

registro y ver si satisface una condición particular. No importa la secuencia del archivo, ya que lo que se quiere ver es si los datos almacenados cumplen o no ciertas condiciones impuestas.

Supongamos un almacén de productos farmacéuticos controlado por un ordenador. Si quisiéramos saber en un momento determinado cuántos productos están por debajo de una cierta cantidad, utilizaríamos el método de «análisis de contenido». Por ejemplo, supongamos que se desea obtener un listado de productos por debajo de 500 unidades. El proceso consiste básicamente en leer el archivo e ir sacando los productos que estén por debajo de tal cantidad. Para ello se utiliza un archivo de entrada que se compara con un valor constante. Este archivo de procedencia suele estar organizado secuencialmente.

Acceso secuencial-selectivo

El método de acceso Secuencial-Selectivo se utiliza par actualizar un archivo maestro de Entrada-Salida. Para realizarlo se necesitan dos archivos: el maestro y el de transacciones, ambos clasificados de la misma forma. Su aplicación más importante es la actualización de archivos de actividad baja, o, lo que es lo mismo, archivos en los que hay que cambiar pocos datos o registros. Los archivos de gran actividad utilizan la técnica Padre/Hijo.

Veamos como se accede a un archivo mediante la técnica Secuencial-Selectiva. Supongamos un almacén con piezas de recambio para televisores en blanco y negro controlado por un ordenador. Debido a que la televisión en color ya está muy difundida, habrá poca actividad en este almacén. Supongamos además que el archivo sólo contiene tres bloques, cada bloque con cuatro registros. Se retiran una serie de piezas y se forma el archivo de transacciones que afecta a los registros 5, 7 y 11 del archivo maestro.

La actualización se realiza de la siguiente forma: se lee un bloque de registros y luego se escribe si ha sufrido cambios. Un indicador de actividad señala cuando un registro de un bloque se ha actualizado y es preciso escribirlo de nuevo junto con todo el bloque. Este indicador se pone en ON cuando indica actividad y en OFF si no la tiene.

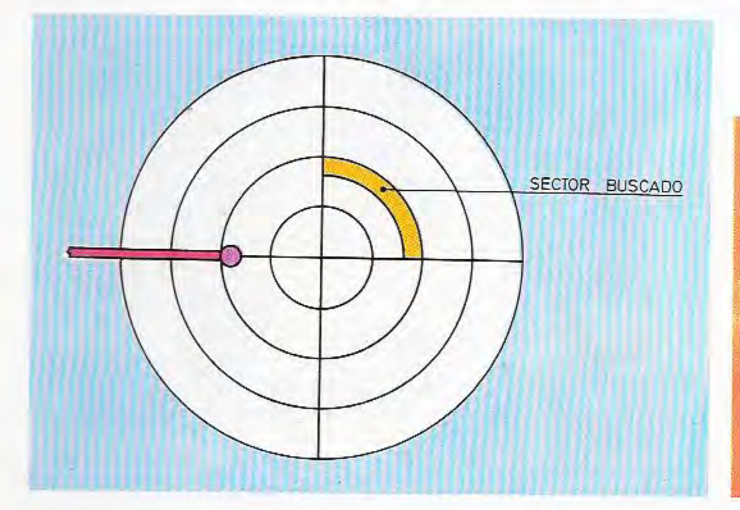
En el ejemplo, con el primer bloque estaría en OFF, puesto que no hay que actualizar ninguno de sus registros. Con los bloques 2 y 3 se colocaría en ON, ya que tienen que actualizarse en sus registros 5, 7 y 11, respectivamente. Conforme se van actualizando, el indicador cambia de ON a OFF para indicar que se ha completado la operación.

Este método tiene algunas ventajas sobre otros métodos de acceso, como el Padre/Hijo, que son:

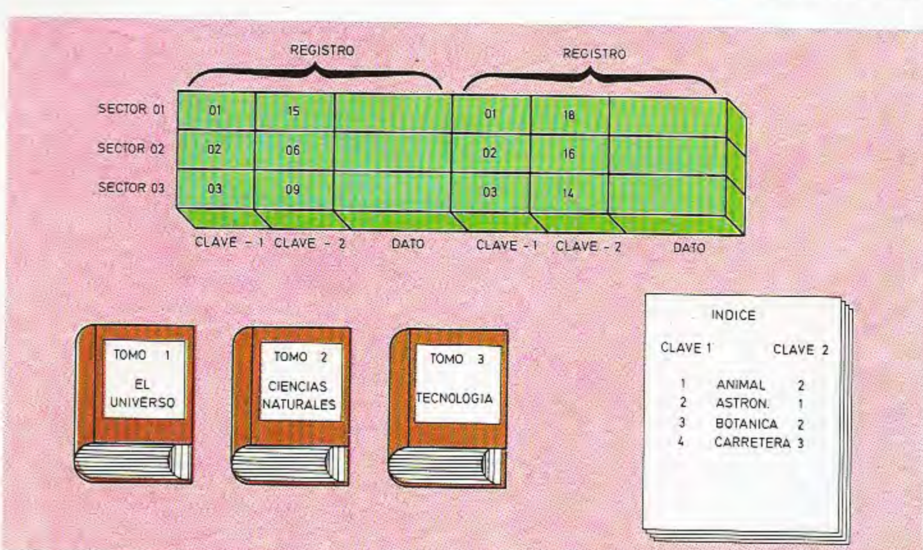
- Una reducción del tiempo de proceso debido a que sólo se reescriben los bloques activos.
- No se necesita un nuevo archivo para la actualización, sino que ésta se realiza en el mismo archivo maestro.

Tiene también una serie de desventajas, como son:

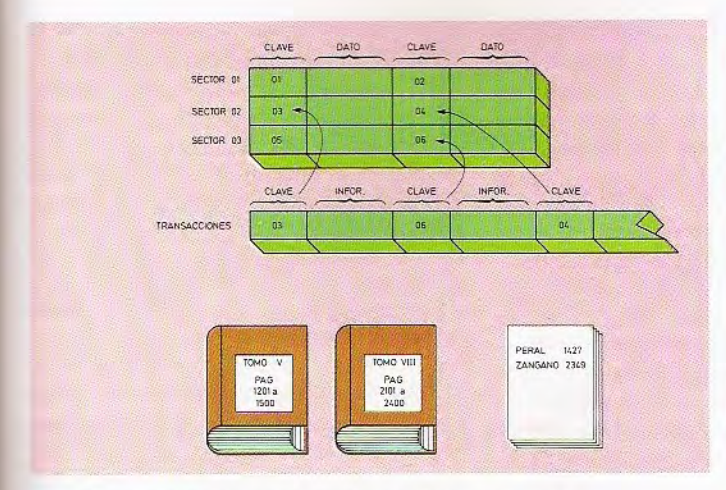
- Sólo se pueden utilizar como medios de almacenamiento el disco y el tambor, ya que estos medios manejan archivos de Entrada/Salida.



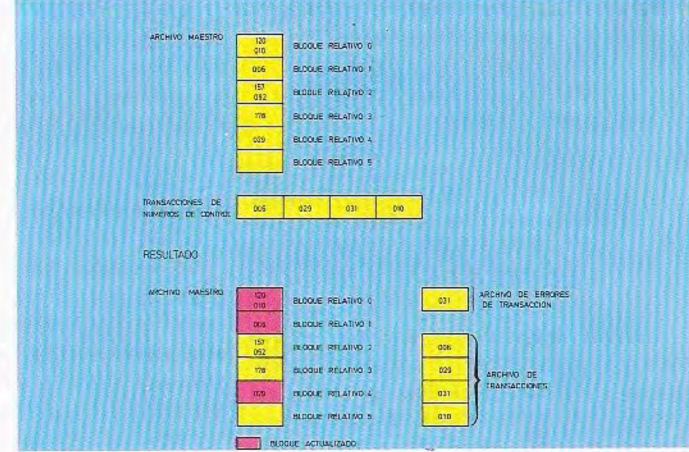
El acceso a los archivos almacenados en disco es casi instantáneo, dado que la cabeza lectora se posiciona directamente sobre la pista donde se encuentra el sector buscado.



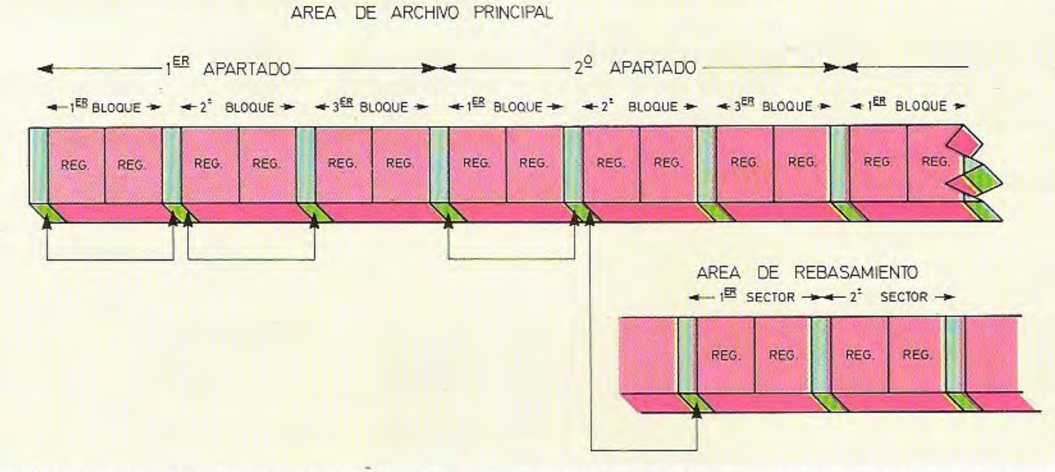
El direccionamiento directo en modo real utiliza la misma técnica que los índices alfabéticos generales de una enciclopedia: la clave 1 indica el número del tomo y la clave 2 el lugar alfabético.



En el direccionamiento relativo, la organización es semejante a la de una enciclopedia, en la que la numeración de las páginas prosigue de un tomo a otro.



Cuando un archivo se actualiza por el método UCA, la clave elegida para designar a cada bloque es el resto de dividir por cinco el número de cada registro. El bloque 5 se suele emplear como área de rebasamiento.



En determinados casos resulta necesario introducir nuevos datos en un bloque de información ya completo. Para solventar este problema se utiliza la denominada «área de rebasamiento» que constituye la zona en la que se colocarán los nuevos datos.

Organización encadenada

La organización encadenada es otro método de organización de archivos consistente en estructurar los bloques del archivo para que se puedan encadenar entre sí automáticamente.
Los bloques de los registros consecutivos no tienen que estar juntos unos con otros, sino que es el sistema operativo el que los enlaza entre sí de forma automática.
El encadenamiento se realiza con ayuda de los punteros. Un puntero es un campo de referencia incorporado a un registro o bloque que contiene una referencia para saber a qué registro o bloque hay que acceder a continuación. Por medio de punteros sucesivos se pueden enlazar registros o bloques de registros.

Rebasamiento de un medio de archivo

Se define como área de rebasamiento de un medio de archivo magnético a la parte del archivo que permite su ampliación.
Supongamos que un archivo está completo y necesitamos incorporarle nuevos datos; la única forma de hacerlo es situando estos datos en una zona reservada previamente para este fin; esta zona es la que se denomina área de rebasamiento.
Para utilizar estas áreas se necesitan ciertos métodos para relacionar perfectamente la parte propia del archivo y la del rebasamiento.
La organización secuencial no admite áreas de rebasamiento, pero sí la organziación de archivos encadenados.

En la figura adjunta se representa una SECCION de un archivo encadenado, con sus sectores, bloques y registros. Esta sección se divide en dos zonas: el área de archivo principal y el área de rebasamiento. Vamos a enlazar bloques de registros en lugar de registros. El área de rebasamiento del archivo va a aceptar datos (registros) que no pueden ocupar, por falta de espacio, cualquiera de los bloques del archivo principal.

- No se pueden eliminar o insertar registros durante la actualización, ya que el bloque se vuelve a escribir en el mismo espacio que contenía el bloque primitivo.

Tratamiento de archivos directos

El acceso directo a un archivo se diferencia básicamente del acceso secuencial en el tiempo necesario para llegar a un determinado registro. En el acceso directo este tiempo no depende, en absoluto, de la situación del dato dentro del archivo; se puede decir que el tiempo que se tarda en acceder a cada uno de los registros es prácticamente el mismo.

Existen diferentes tipos de acceso directo; todos tienen en común la utilización de una técnica de direccionamiento por la que se puede localizar un registro, o el bloque en el que se encuentra el registro, sin tener que acceder a ninguno de los otros registros o bloques del archivo.

El acceso directo en archivos de Entrada/Salida se emplea, normalmente, para procesos en tiempo real, es decir, para procesos que requieren mucha velocidad de respuesta. Al acceso directo se le conoce también con el nombre de acceso al azar.

Existen dos métodos básicos de acceso directo: el método de direccionamiento directo y el de direccionamiento por fórmula, conocido también por UCA:

- *Direccionamiento directo*

Como siempre, tendremos dos archivos: el archivo maestro y el archivo de transacciones. En este método la clave (o parte de la clave) del registro de transacción representa la situación del registro correspondiente del maestro. Este sistema actúa de dos modos:

– En el modo *dirección real*, toda la clave o parte de ella es la dirección de la pista o sector del disco en el que se encuentra el registro del archivo maestro. Normalmente los archivos tienen varios registros por bloque, pero en el caso de que sólo haya uno y de que cada bloque se encuentre en un sector, lo único que necesitamos conocer es la clave correspondiente de la dirección del sector.

Veamos un ejemplo en el caso de que el archivo tenga varios registros por bloque. Supongamos un archivo maestro de tres sectores, con dos registros cada uno de ellos, y cada registro con dos claves. La primera de ellas indica el sector en el que se encuentra el registro y la segunda identifica el lugar del registro dentro del archivo. Los dos registros del sector 01 tienen entonces como primera clave 01, pero la segunda clave ya no está ordenada. Cuando se quiere actualizar el archivo maestro, las transacciones deben tener ambas claves.

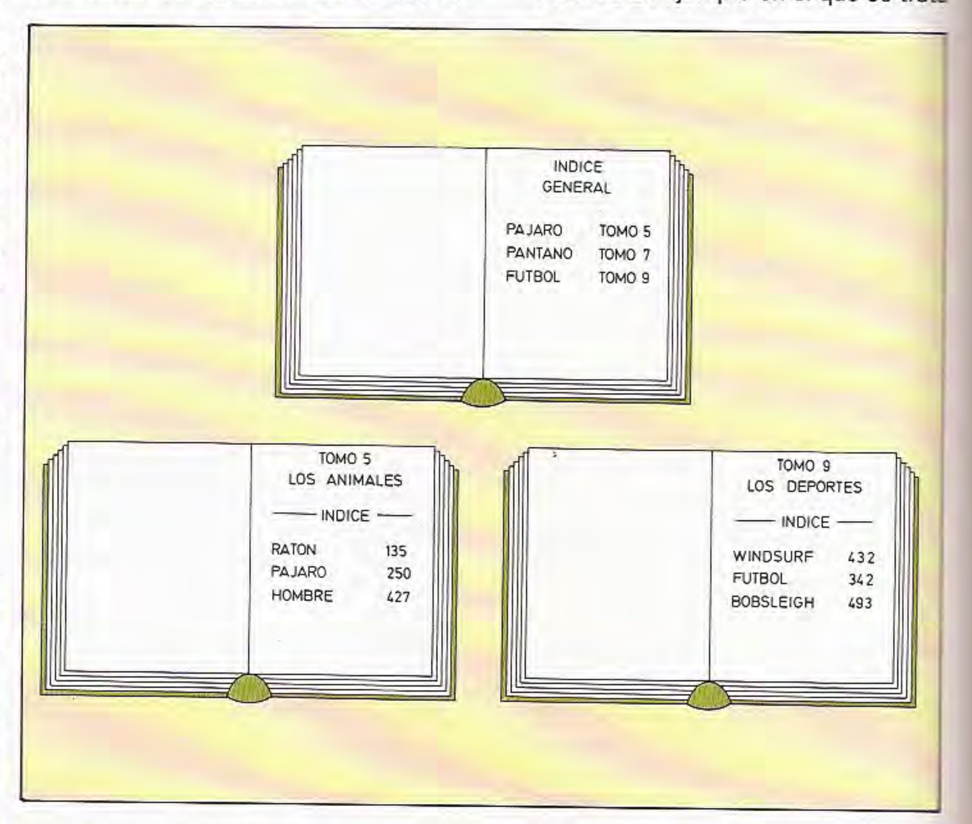
– En el modo de direccionamiento relativo, se accede a los registros maestros indicando la posición relativa del registro. El registro de transacciones contendrá en un campo de clave la posición relativa de los registros. Las transacciones pueden estar clasificadas o no, como en el caso anterior, con la clave 2.

Se observa que utilizando el método de direccionamiento directo sólo hay que procesar los registros del maestro que se vean alterados por los del archivo de transacciones. Se pueden emplear transacciones clasificadas o no, pero no es muy rentable desde el punto de vista de tiempo cuando el archivo maestro tiene mucha actividad.

- *Direccionamiento por fórmula (UCA)*

Este método tiene la particularidad de que cada registro del archivo maestro se puede localizar mediante una sola clave utilizando un algoritmo o fórmula matemática para calcular la dirección del registro. El archivo se creará mediante una rutina que contendrá el programa de aplicación para obtener, mediante la fórmula matemática, la posición del registro.

Veamos un ejemplo en el que se trata



Un archivo secuencial indexado utiliza un directorio parecido al índice de un libro. De igual forma que un índice general puede remitir al índice de un tomo, el directorio puede, igualmente, tener varios niveles de clasificación.

de dividir por cuatro el campo de identificación para determinar la dirección del registro. Tenemos un archivo de tres bloques con dos registros cada uno. Puede surgir un problema en esta forma de cálculo, que es el siguiente: supongamos que la clave de identificación sea 04, al dividirla por 4 nos da que el registro de clave 04 se grabará en la pista 1. si la clave siguiente de identificación fuera 06, al dividirla por 4, también nos daría 1 y se grabará, igualmente, en la pista 1, pudiendo aparecer problemas de espacio. Para evitar esto se dejarán unos espacios sin grabar para que se puedan almacenar los datos que no quepan en la pista asignada. A esto es a lo que se llaman registros SINONIMOS (registros que podrían ocupar la misma posición).

Se pueden emplear diferentes tipos de fórmulas (fórmula del número primo, del cuadrado, etc.). Este método puede ocuparse de transacciones sin clasificar, tiene una respuesta rápida y sólo se procesan los registros del maestro afectados.

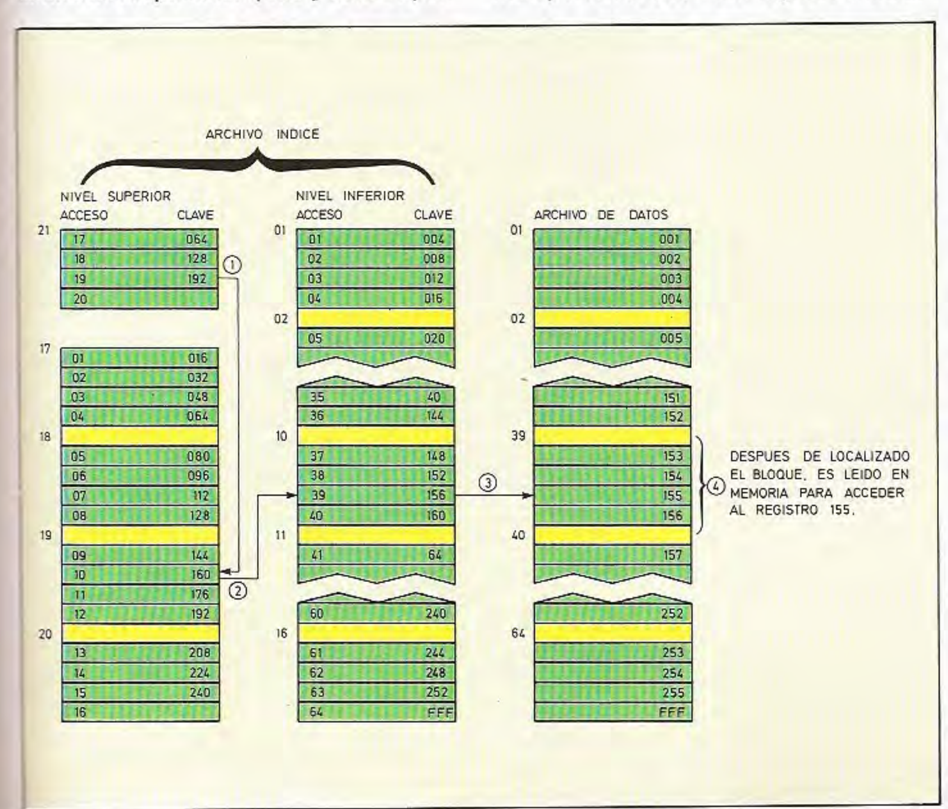
Tratamiento de archivos indexados

El archivo indexado elimina dos de los defectos principales de los archivos directos:

- Evita la conversión de cada clave de identificación en una clave relativa.
- No hay que guardar en memoria las posiciones de registros que no se utilizan en el archivo.

Mediante la organización indexada se puede acceder a un archivo de forma directa o de forma secuencial.

Veamos cómo funciona este tipo de archivos. Cuando se ejecute una instrucción de acceso al azar (directo), el sistema operativo localiza el registro median-



Ejemplo de un sistema secuencial indexado de tres niveles. El archivo índice de orden superior remite a un nivel intermedio a través del cual se llega a un nivel inferior. En este último se localiza la posición del registro buscado dentro del archivo de datos.

Para saber más

¿A qué se llama archivo maestro?

Se llama archivo maestro a aquel que contiene una información básica que cambia muy poco. Por ejemplo, un archivo que contiene los datos de los empleados de una empresa es un archivo maestro. En general, sólo se modifica ocasionalmente para dar altas, bajas, cambios de categoría, etc.

¿Qué es un archivo de transacciones?

Se denominan archivos de transacciones aquellos que contienen datos que sirven para procesos de cálculo o de actualización de los archivos maestros.
Un archivo con las horas trabajadas en una semana por los empleados sería un archivo de transacciones.

¿Qué diferencia hay entre Fichero y Archivo?

Ninguna. Son dos términos sinónimos. Ambos son traducciones de la palabra inglesa «File». Algunos autores distinguen entre archivo (concepto abstracto) y fichero (el conjunto físico).

¿Qué diferencia hay entre una sección lógica y una física?

La sección física viene impuesta por el medio, mientras que la lógica depende de la organización del archivo. En una organización clásica, una sección física sería un archivador, mientras que una sección lógica serían las facturas de un año determinado.
Es evidente que en un archivador (sección física) pueden almacenarse facturas de varios años (varias secciones lógicas). No obstante, si las facturas son numerosas podrían ser necesarios varios archivadores (secciones físicas) para guardar las facturas de un solo año (una sola sección lógica).

¿Por qué se agrupan los registros en bloques?

Las operaciones más lentas son las entradas y salidas, puesto que implican el uso de medios electromecánicos. Si se agrupan las informaciones en bloques disminuye el número de operaciones de entrada/salida y, por tanto, disminuye el tiempo de proceso.

te una búsqueda en el DIRECTORIO DE INDICES DE CLAVE. El archivo indexado se organiza de tal forma que cada registro se identifique por una clave llamada CLAVE DE REGISTROS, que se incluye en el directorio de índices de claves. La clave no indica directamente la posición del registro en el archivo, lo único que hace es identificarlo.

El directorio de índice de clave es un archivo creado por el sistema operativo del ordenador cuando se crea el archivo maestro. Contiene las claves de los registros, en orden ascendente, y la posición de los registros del archivo maestro.

Cuando el tamaño del archivo es muy grande, el índice se puede colocar en niveles que reducen el número de claves que hay que examinar al acceder al archivo. Se pueden utilizar hasta tres niveles. Cada registro del nivel superior hace referencia al último registro de un bloque del nivel intermedio. Un registro del nivel intermedio hace referencia a otro del nivel inferior y este último llega al registro del archivo maestro.

Cuando el directorio de índices tiene un solo nivel y los registros del archivo maestro están ordenados secuencialmente, se tiene un archivo SECUENCIAL INDEXADO. Cuando se quiere encontrar un registro determinado del archivo maestro, se efectúa una búsqueda en el índice del archivo para determinar su posición. La forma de hacer esta búsqueda es muy parecida a la de buscar una palabra en un diccionario. Vamos a ilustrar este método de acceso mediante un ejemplo. Sea una tabla con una serie de palabras CLAVES de comienzo de página y el número de página. Supongamos que se quiere encontrar la palabra COCHE; tendremos que buscar en qué página se encuentra la clave en secuencia alfabética igual o mayor y esta palabra clave es CIBERNETICA. En la página de CIBERNETICA se encontrará la palabra COCHE, sólo hay que buscarla leyendo en forma secuencial esta página.

El archivo SECUENCIAL INDEXADO sustituye el número de página por las direcciones de los sectores del disco y las palabras claves por las claves del registro, pero todo lo demás es igual a lo visto en el ejemplo anterior del diccionario.

La gran ventaja de la organización indexada es que el sistema operativo es el que mantiene el índice y el programador no necesita crearlo.

En el modo de acceso secuencial se accede a los registros en secuencia ascendente del valor de la clave del registro. En el acceso directo, la secuencia de acceso es controlada por el programa. Cabe precisar que una de las desventajas de la organización indexada es que como el índice está almacenado en disco, es preciso más de un acceso al disco para leer y buscar el índice, mientras que en la organización directa esto se hace en un solo acceso. En el cuadro-resumen se ven las distintas organizaciones de archivo con sus medios de acceso y de almacenamiento.

Acceso particionado

En los ficheros de *acceso particionado* (PAM), los registros se agrupan en «miembros». Cada miembro se identifica con un nombre que se encuentra al principio del archivo, en un espacio reservado que se llama directorio. Junto al nombre del miembro aparece la dirección de comienzo. Los distintos miembros se graban uno detrás del otro según llegan.

El acceso se efectúa de la siguiente forma: a partir del directorio se localiza el miembro al que se quiere acceder de forma directa y, a continuación, de forma secuencial, se accede a los registros del miembro en cuestión. Este método de acceso se utiliza fundamentalmente para almacenar programas.

Acceso virtual

Con el método de *acceso de memoria virtual* se puede acceder a ficheros secuenciales indexados y directos. Utiliza el llamado intervalo de control, que es el elemento de transmisión entre la memoria auxiliar y principal. Este se compone de un número entero de registros o bloques y tiene una longitud fija múltiplo de 512 bytes. Existe también un área libre que se reserva para futuras ediciones del fichero.

Para saber más

¿Se podría grabar un archivo en cinta sin agrupar los registros por bloques?

Sí se podría, pero no sería muy operativo, ya que los tiempos de acceso serían muy grandes al querer acceder a un registro o registros determinados. También la ocupación efectiva de la cinta sería muy baja.

¿Cuándo se emplea, fundamentalmente, la división de archivos por secciones?

Se emplea cuando se tiene que organizar un archivo multicarrete. La información grabada en cada carrete es la que recibe el nombre de sección. El programador debe preparar el programa que procese este tipo de archivo para que automáticamente pueda pasar de una cinta a otra cuando se llega al final de la sección. Este procedimiento se conoce con el nombre de «enlace de sección a sección».

¿Qué es un «cartucho de cinta magnética»?

Es un medio de almacenamiento de archivo parecido al casete de cinta magnética, pero con una capacidad de almacenamiento mucho mayor.

¿A qué se denominan copias de seguridad de archivos?

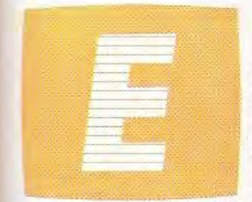
Si se grabara una determinada información en un solo medio de archivo, podría ocurrir que, debido a cualquier fallo del sistema de proceso, se borrara parte de la grabación, con la consiguiente pérdida de información. Para evitar esto normalmente se hacen dos grabaciones de un mismo archivo. Una de ellas se utiliza para ser procesada y la otra se guarda con fines de seguridad. Esta última, llamada copia de seguridad, se graba en cinta magnética, tanto si la grabación original se hizo en cinta como si se hizo en disco, ya que la cinta es mucho más barata que el disco.

¿Las secciones tienen que corresponder siempre a un carrete?

No. A veces conviene crear en un solo carrete (sección física) varias secciones (secciones lógicas).

Métodos de proceso de datos

Rentabilizando los recursos del ordenador



En los primeros años de la informática los programas se ejecutaban uno a uno, de forma que un solo proceso ocupaba a toda la unidad central hasta que se daba por terminado. Este método de proceso de datos, denominado de *secuencia simple,* tiene el inconveniente de mantener inactivo al microprocesador mientras se gestionan las operaciones de entrada/salida; operaciones que son, además, las que tardan más tiempo en ejecutarse.

El intento de tener ocupada la unidad central durante el máximo tiempo posible dio lugar a la aparición de nuevos métodos de proceso que se pueden clasificar según dos criterios. Atendiendo al tiempo que se invierte en realizar las operaciones de entrada/salida y de proceso, nos encontramos con los procesos en *lotes* y en *tiempo real.* Si reparamos en el lugar donde se van a procesar los datos, habrá que diferenciar entre sistemas de proceso *centralizados* y *descentralizados.*

Proceso de datos en lotes

Cuando una empresa es pequeña, un solo empleado puede realizar todas las operaciones de contabilidad, suministro de pedidos, etc. En el caso de una empresa grande, es evidente que un empleado no podría realizar todos los procesos que ésta necesita para un eficaz funcionamiento. La empresa en cuestión optará normalmente por dividir el trabajo en *«lotes».* La parte de contabilidad, por ejemplo, la realiza un empleado o máquina; de las transacciones de pedidos de almacén se encarga otro, etc. El proceso por *lotes* es, en este caso, más económico que el de ejecutar todas las operaciones a la vez. Veamos a continuación tres técnicas de proceso de datos en lotes:

- *Proceso en lotes de acceso al azar*

Es muy adecuado para tratar ficheros de baja actividad. Sólo se accede a los registros maestros que se vayan a procesar. Los registros actualizados se graban en el mismo espacio en el que se encontraban los originales. Las transacciones no tienen que clasificarse, lo que ahorra tiempo de procesamiento. Es la misma técnica que utiliza el acceso de archivos directos.

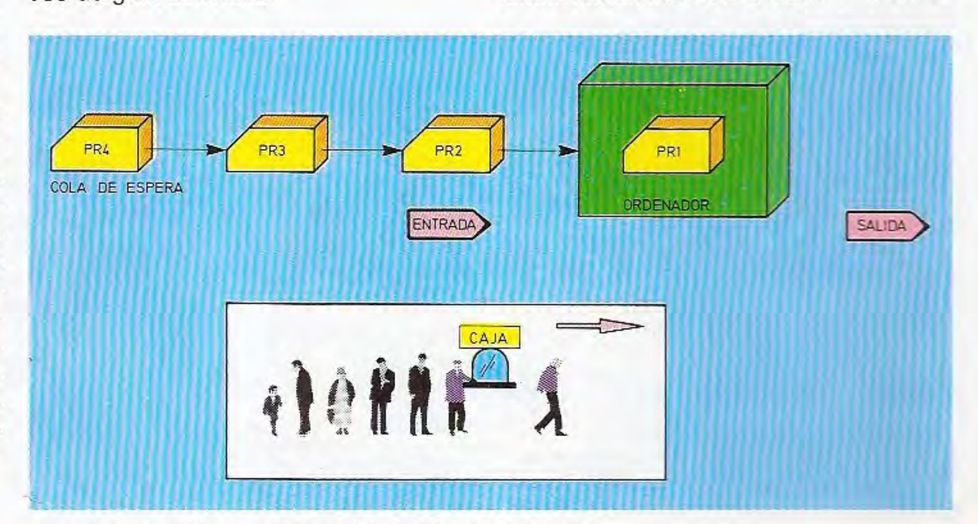
- *Proceso en lotes Padre/Hijo*

En capítulos anteriores se ha analizado ya esta técnica. Se utiliza para archivos de gran actividad.

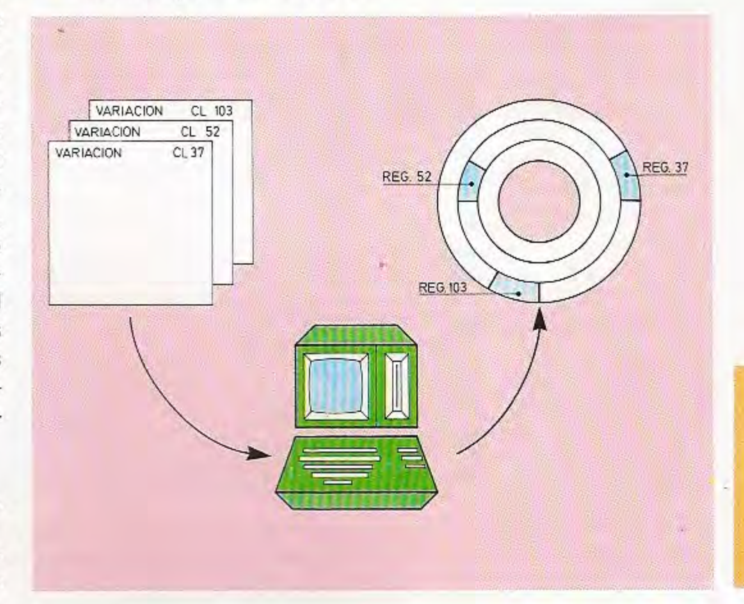
- *Proceso en lotes remoto*

Muy utilizado cuando tenemos un ordenador central comunicado con varios terminales que pueden estar alejados del mismo. Veamos la filosofía de este proceso utilizando un ejemplo.

Supongamos una empresa que suministra recambios de automóviles. Esta empresa tiene repartidas una serie de sucursales; por ejemplo, cuatro, en lugares geográficamente distanciados de la sede central. Estas sucursales tienen unos terminales conectados al ordena-



En el método de secuencial simple, un programa no empieza a procesarse hasta que ha finalizado la ejecución del anterior. El ordenador se comporta como el funcionario de una oficina pública en la que los clientes, o programas, se sitúan en la fila para ser atendidos.



En el proceso por lotes de acceso al azar, el ordenador sólo accede a los registros del archivo maestro que deben actualizarse.

dor del almacén central. Su forma de trabajo es la siguiente:

Durante la jornada de trabajo, cada sucursal va coleccionando los datos de las piezas que vende en un dispositivo llamado «colector de datos».

Al final de la jornada, un programa ubicado en el ordenador central va «preguntando» a cada sucursal por el número de ventas realizadas y actualiza su archivo central de almacén, controlando el número de piezas que le quedan en stock y las que les queda a sus sucursales. Si fuera preciso, el almacén central mandará a las sucursales las piezas que les sean necesarias.

Este proceso tiene muchas ventajas, ya que ni hay que utilizar el teléfono ni que recurrir a correos para la comunicación de las sucursales con la central. Por otra parte, el control de las ventas es total.

Proceso de transacciones

Es un método muy utilizado en la actualidad. Cada transacción se procesa totalmente, y sus archivos se actualizan de una vez. Utiliza archivos de acceso al azar, con organización relativa o indexada. Se diferencia del proceso por lotes al azar en la reducción a tiempos muy bajos del ciclo de la transacción, y en que la actualización de archivos se hace inmediatamente. El software y las técnicas de programación utilizados permiten el procesamiento de las transacciones siempre que lo requiera el usuario.

Proceso interactivo

Este método se fundamenta en un diálogo constante entre el programa de ordenador y el operador. El instrumento para este diálogo suele ser la pantalla y el teclado.

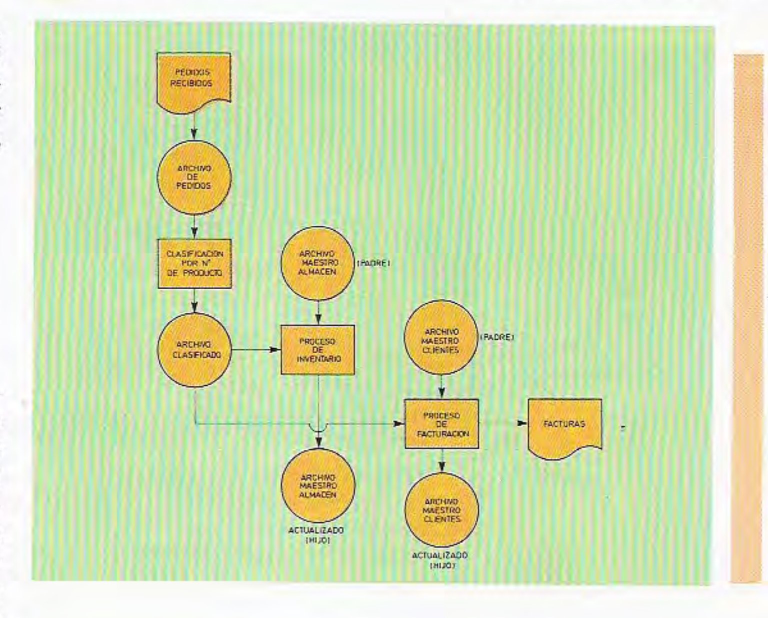
Para explicar este proceso vamos a utilizar un ejemplo. Suponga que el operador que trabaja con los archivos de clientes y productos, recibe el pedido de un cliente.

Usando la pantalla, el operador comprueba que ese cliente es nuevo, ya que no está en el archivo de clientes. El programa le pregunta si hay que pasar al nuevo cliente al archivo de clientes. El operador responde que sí y la pantalla se prepara para recibir el nombre, dirección, etc., del nuevo cliente. Después, el programa le pregunta si la información es correcta, y el operador la revisa en pantalla y dice que sí. El software le pregunta que más quiere hacer, y el usuario responde que introducir un pedido... Todo un proceso de constante diálogo: «interactivo».

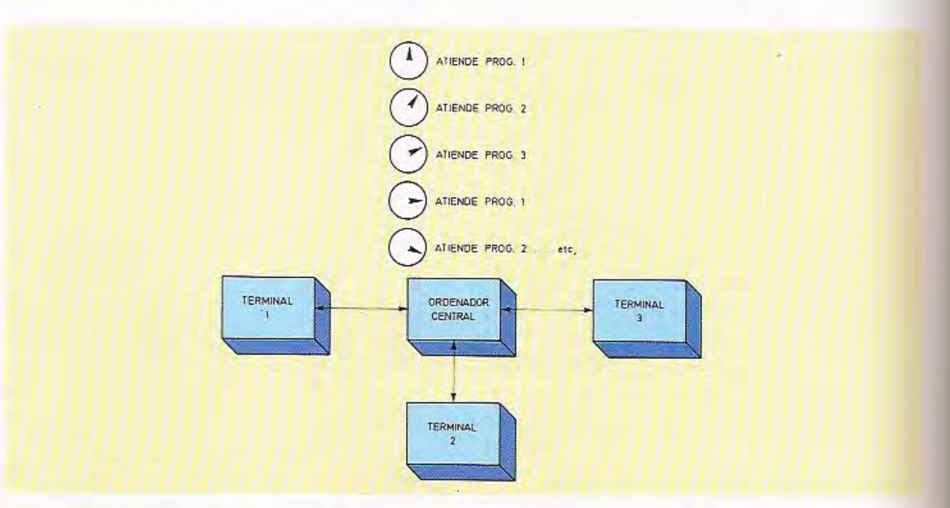
Vemos que con este método el desarrollo normal del programa se altera de acuerdo a los datos de entrada. Es un método que relaciona de una forma directa la entrada de datos con los archivos maestros.

Proceso en tiempo real

El proceso en tiempo real requiere que exista conexión entre los terminales y el ordenador central. El proceso tiene que ser lo suficientemente rápido como para que el resultado de una operación tenga



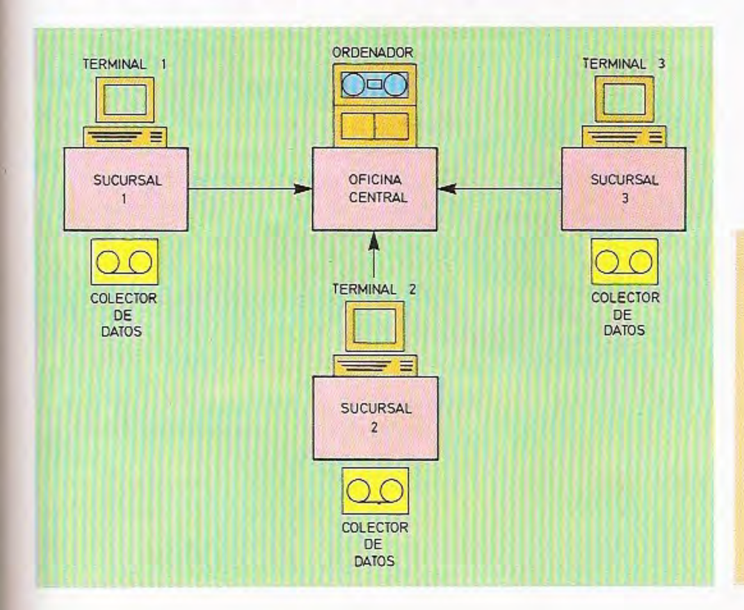
Ejemplo de actualización de los datos de un almacén, utilizando el método padre/hijo. Al recibir un pedido los archivos padre de clientes y de almacén se actualizan, mediante los procesos de facturación e inventario, en sus correspondientes archivos hijo.



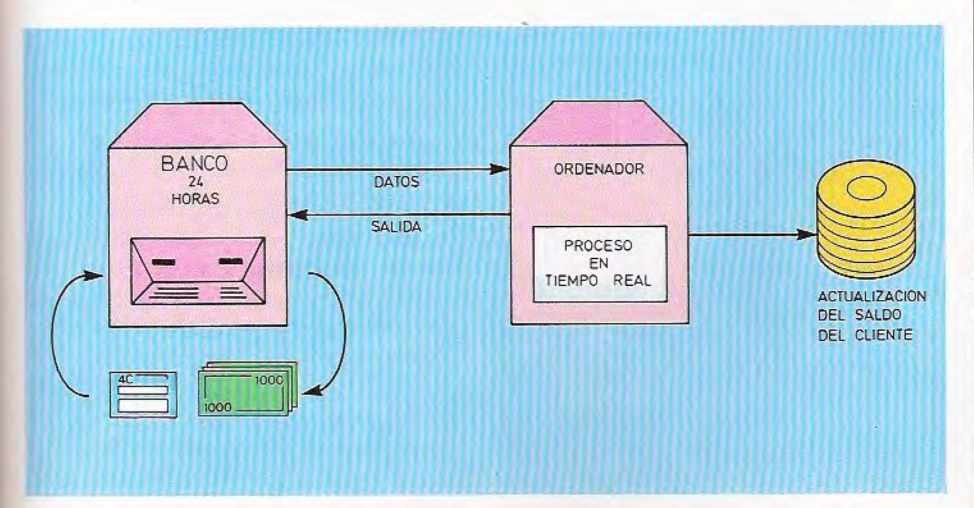
El proceso en tiempo compartido reparte la capacidad de trabajo del ordenador central entre sus diversos usuarios, quienes, de esta forma, tienen la sensación de ser los únicos en utilizar el ordenador.

un efecto inmediato sobre el proceso que se está realizando. Un ejemplo típico de proceso en tiempo real surge cuando llegamos a un terminal cajero de un banco y queremos sacar dinero. Tecleamos nuestro número de cuenta y pedimos una cantidad (10.000 pesetas, por ejemplo). Antes de entregarnos el dinero, el ordenador comprueba en el fichero de cuentas de clientes el número tecleado y si disponemos de saldo suficiente. Si todo es correcto, dará las órdenes oportunas al cajero para que nos las dé. En caso contrario, le dirá que no nos dé nada y que nos informe que la operación no es correcta. El ordenador central puede estar atendiendo a la vez a varios terminales: mientras está buscando o actualizando nuestro saldo puede estar ejecutando otro tipo de operaciones con otros cajeros.

En la actualidad, los ordenadores centrales pueden operar con procesos en *lotes* y en *tiempo real.* Cuando el ordenador central no recibe una demanda de un terminal de tiempo real, puede estar ejecutando un proceso en lotes. Cuando aparece esta demanda interrumpe el proceso en lotes para atender al terminal.



En el proceso por lotes remotos los datos almacenados en cada sucursal se transmiten al ordenador central, en el que se acumulan para su posterior tratamiento.



El proceso en tiempo real utiliza archivos de acceso directo; la actualización de los archivos es inmediata y el tiempo de respuesta mínimo.

Proceso en línea (on line)

El proceso en línea es aquel en el que un terminal situado a una distancia lejana de un ordenador central se comunica con él. Ya hemos visto algunos procesos en línea como el de lotes remoto y el de tiempo real, pero aún existen otros tipos que merecen nuestra atención.

- *Sistema de consultas*

Este proceso se utiliza para pedir información o «consultar» al ordenador central. Ejemplo típico de este sistema es la consulta sobre los créditos disponibles que efectúa una sucursal bancaria al ordenador central. En el caso de ventas, otro ejemplo es la consulta que puede hacer un punto de venta sobre las existencias de material en el almacén central. Hay que indicar que este método realiza la actualización de archivos como lo hacía el proceso en lotes, es decir, al final de la jornada de trabajo.

- *Conmutación de mensajes*

Un usuario puede enviar un determinado mensaje a uno o más terminales remotos. Un ejemplo típico es el de un vendedor que ordena el envío de pedidos a uno o más almacenes.

- *Captura de datos*

Los terminales de varias sucursales envían al almacén central varios pedidos que quedan almacenados en un archivo para procesarlos en el momento que desee el ordenador central.

- *Sistemas de actualización de archivos*

En este caso los datos mandados por los terminales se procesan en el mismo momento en que llegan. Ejemplo típico es la venta de billetes de tren.

- *Sistemas de control de procesos*

Estos sistemas se utilizan en aplicaciones de tipo industrial con fines de supervisión y de control de procesos industriales. En la industria electrónica la fabricación de circuitos es supervisada y controlada por un sistema de este tipo.

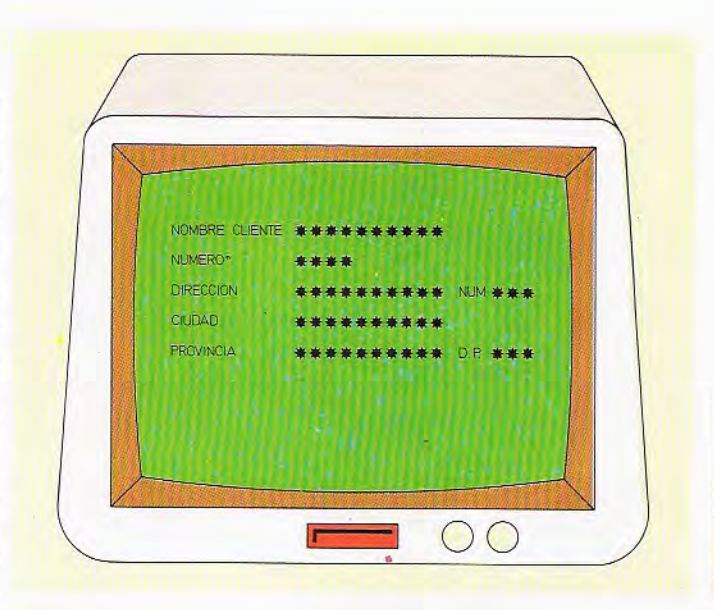
- *Procedimiento de archivo imagen*

Con este proceso los archivos utilizados durante el procesamiento de la jornada no son los reales, sino una copia exacta de ellos. Los archivos imagen se actualizan durante el día y proporcionan la información que se les pida al hacerles una consulta. Los archivos reales se actualizan durante la noche y, a partir de ellos, se preparan los nuevos archivos imagen que se utilizarán en la jornada siguiente.

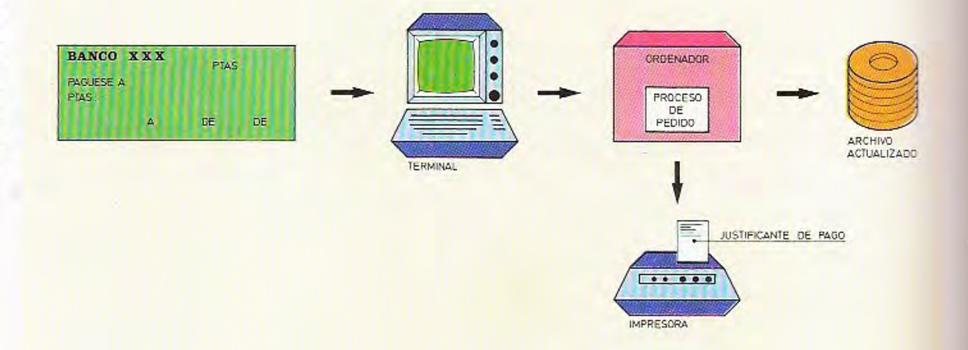
Tiempo compartido

Hay que evitar que un ordenador esté sin ejecutar programas, ya que el tiempo en el que un ordenador está inactivo cuesta mucho dinero. Para solventarlo se utiliza el proceso en tiempo compartido. Supongamos una empresa con un ordenador central y varios departamentos que necesitan utilizarlo. El ordenador central tiene una tabla de prioridades y va atendiendo siempre a los diferentes procesos, de manera que el primero que realiza es el de mayor prioridad, y el último que procesa es el de prioridad más baja.

Actualmente, el sistema en tiempo compartido funciona de la siguiente manera. Un ordenador central está conectado a una serie de terminales. Este ordenador, mediante las técnicas de multiprogramación, asigna un determinado tiempo a cada terminal para que pueda ejercer su derecho a realizar el proceso. Una vez que ese tiempo se cumple, el ordenador central se desconecta de ese terminal y pasa a otro. Esto se hace tan rápido que a cada usuario le parece ser el único que utiliza el ordenador central.



En los procesos interactivos se establece un diálogo entre el ordenador y el operador, a través de la pantalla.



En el proceso de transacciones se utilizan archivos de acceso directo. Se evita, con ello, la necesidad de clasificar documentos y la actualización de los registros es inmediata.

Proceso distribuido

En los procesos descritos hasta ahora se utiliza un ordenador central para realizar el proceso de los datos. El sistema distribuido elimina el ordenador central: las funciones de proceso la realizan unidades ubicadas en diferentes estaciones de proceso. Esto implica utilizar una serie de ordenadores y terminales conectados entre sí por una red de comunicaciones.

Proceso descentralizado

Es parecido al sistema distribuido, pero con la diferencia de no utilizar sistema de comunicaciones. En este caso cada departamento de una empresa tiene su propio equipo de proceso y el personal adecuado para su operación.

Para saber más

¿Qué significa proceso «BATCH»?

La palabra inglesa BATCH es el nombre con el que se conocen, en el campo profesional, los procesos por lotes.

¿Qué significa proceso «BACKGROUND»?

De nuevo nos encontramos con terminología inglesa. En el campo profesional los procesos BACKGROUND son los de baja prioridad que se ejecutan, de forma automática, cuando los de mayor prioridad no están utilizando los recursos del sistema.

¿Qué quiere decir proceso «FOREGROUND»?

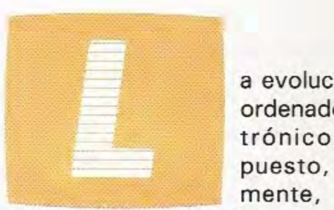
Es la ejecución automática de los programas de más alta prioridad, generalmente en tiempo real. También se les llama de programa preferente.

¿Qué otros significados tiene el proceso «ON LINE» (en línea)?

Cuando una persona se comunica directamente con el ordenador central, sin la utilización de tarjetas perforadas o cinta magnética, se dice que está «en línea». Cualquier unidad periférica preparada para conectar directamente con el ordenador central por sí misma se dice que está «en línea».

El sistema operativo

La inteligencia esencial de la máquina



La evolución de los ordenadores electrónicos ha supuesto, paralelamente, una gran complicación en la lógica de su funcionamiento. Para conseguir un uso más racional y un mejor aprovechamiento de los ordenadores se han desarrollado una serie de programas que constituyen el software_funcional. Los constructores de ordenadores lo suministran bajo diversos nombres, aunque el más general es el de *Sistema Operativo.*

Existen diversas definiciones de Sistema Operativo, aunque nosotros seguiremos la del AUERBACH EDP Report.

«Sistema operativo es una colección ordenada de rutinas y procedimientos que acompañan al ordenador y que normalmente realizan todas o algunas de las siguientes funciones:

- Planificación, carga, inicialización y supervisión de la ejecución de programas.
- Gestión de memoria, unidades de entrada/salida y otros dispositivos.
- Inicialización y control de todas las operaciones de entrada/salida.
- Tratamiento de errores.
- Coordinación de las comunicaciones entre el sistema y el operador.
- Mantenimiento de un registro con las operaciones del sistema.
- Control de las operaciones en los trabajos de multiprogramación, multiproceso y tiempo compartido.»

En resumen, el sistema operativo es el conjunto de los programas del sistema que permiten al usuario utilizar la máquina cómodamente y que optimizan su rendimiento. Una característica fundamental del sistema operativo es que debe incluir un programa *monitor* que controle la ejecución de los restantes programas y mantenga el funcionamiento del ordenador, sin intervención del operador más que en caso de necesidad.

Tipos de sistemas operativos

Podemos distinguir cinco tipos principales:

- *Secuencial por lotes*

Permite ejecutar los trabajos uno a uno. Los programas pueden ejecutarse nada más ser introducidos o memorizarse en dispositivos de acceso rápido, ejecutándose secuencialmente más tarde.

- *Multiprogramación*

Permite que varios trabajos se ejecuten simultáneamente. Se consigue mediante el uso de las interrupciones.

- *Tiempo real*

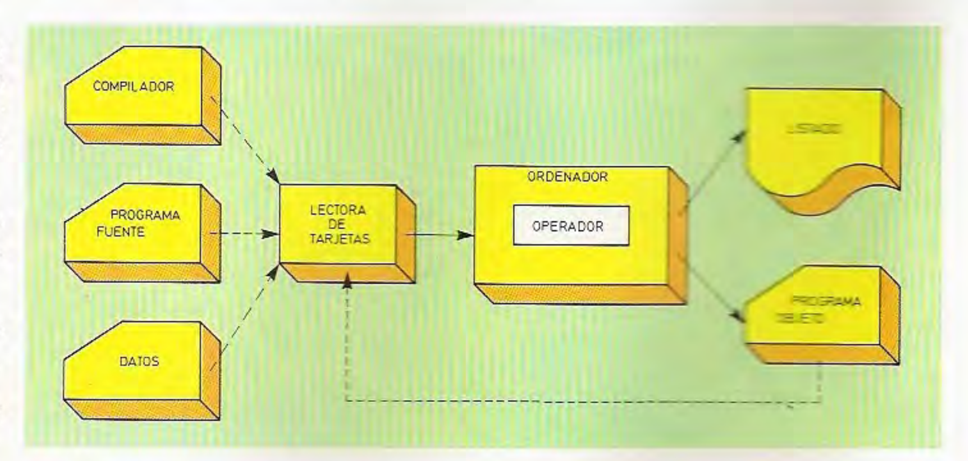
Permite el uso del ordenador por varios usuarios que utilizan terminales remotos y efectúan constantemente operaciones de entrada y salida de datos.

- *Tiempo compartido*

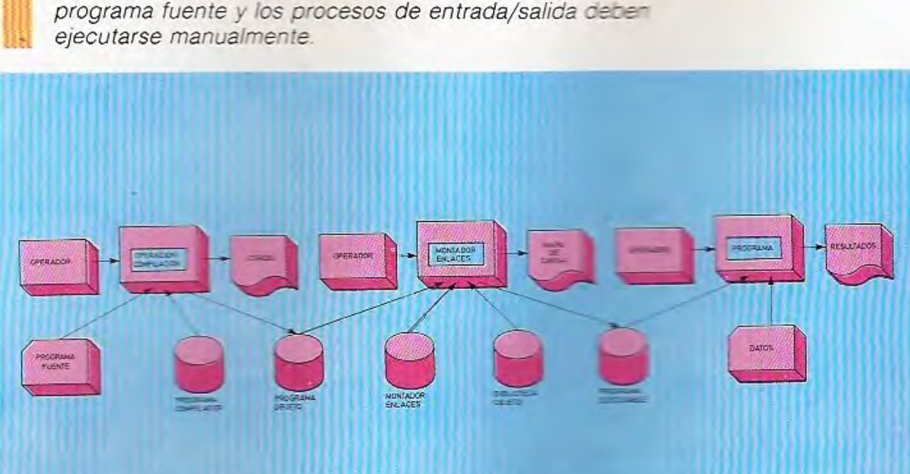
Permite a muchos usuarios utilizar el mismo sistema que, aparentemente, sólo está dedicado a cada uno de ellos, ya que cada usuario recibe el control de la CPU durante un determinado intervalo de tiempo.

- *Memoria virtual*

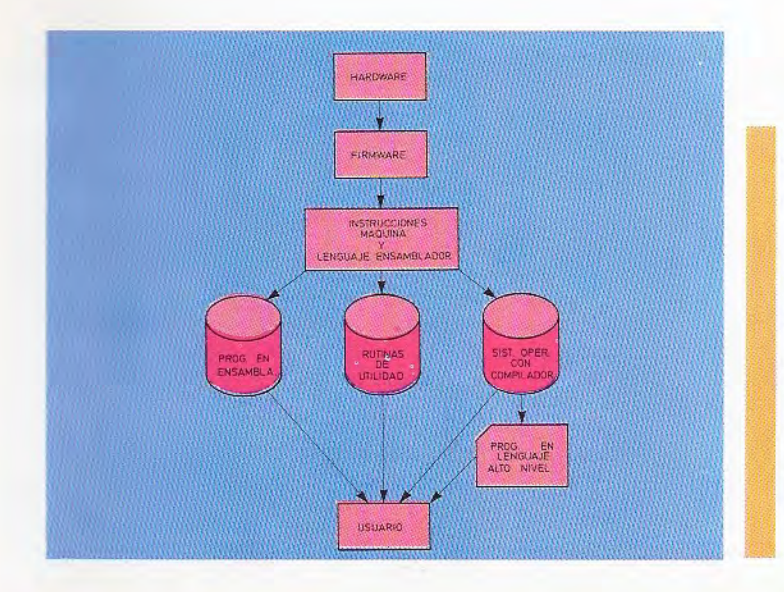
El sistema operativo asume responsabilidades de gestión de la memoria principal.



Explotación manual de un ordenador con lectora de tarjetas y sin sistema operativo. La compilación, la generación del programa fuente y los procesos de entrada/salida deben ejecutarse manualmente.



Cadena de operaciones necesarias para gestionar un programa en un ordenador que utiliza memorias de discos. Las intervenciones del operador resultan inevitables, aunque vienen facilitadas por el empleo de un sistema operativo.



Conjunto de recursos empleados por un ordenador. Entre el usuario y los elementos físicos de la máquina, o hardware, se intercalan todos los procedimientos de microprogramación (o firmware) y de software.

Organigrama o diagrama de flujo de un programa monitor. Cualquier comando ejecutado en un ordenador debe ser supervisado a través de un programa de este tipo.

La implementación práctica de los sistemas operativos se hace mediante técnicas de «overlay» (recubrimiento) u «overlapping» (solapamiento).

Componentes de un S.O.

Los sistemas operativos de la tercera generación además del *monitor* o supervisor, encargado de la gestión de trabajos, contienen componentes que gestionan los recursos del sistema, al propio sistema y a los datos.

La gestión de trabajos se encarga de la organización y regulación del flujo de trabajo en el sistema. También permite que los usuarios se comuniquen con el sistema a través de los comandos del operador y las tarjetas de control.

La gestión de recursos del sistema abarca la asignación al programa seleccionado de los recursos necesarios: memoria, tiempo de CPU, operaciones de E/S, etc. Muchas veces estas tareas las realiza el monitor.

Las funciones de gestión del sistema comprenden la generación del sistema, la conservación del mismo y de los programas, y el interface con los compiladores.

El monitor o supervisor

El monitor o supervisor debe estar presente siempre en la memoria. A veces sólo reside en la memoria central una parte de él, la llamada *residente*. El resto es llamado cuando se necesita.

El supervisor contiene, en general, todos los subprogramas que realizan las funciones básicas. El sistema supervisa la actividad del programa, rechazando las operaciones no válidas y evitando de esta manera la detención del ordenador por los errores del programa de aplicación o programa del usuario.

Los elementos del monitor son: control de trabajos, control de E/S, comunicaciones y recuperación del sistema.

El control de los trabajos comprende las funciones que controlan y regulan el uso de los recursos del sistema y, en

particular, la planificación de trabajos, la asignación de recursos, de carga y la terminación de programas.

El control de E/S regula las actividades de los dispositivos de E/S. Comprende: la planificación de los recursos de E/S, la transferencia de datos y el soporte de los terminales remotos.

El sistema de comunicaciones se responsabiliza de los intercambios de información entre el S.O. y los usuarios. Cuando un error impide la continuación normal de un trabajo intervienen las rutinas de recuperación, que permiten la reanudación de un trabajo a partir de un determinado punto del proceso.

Cuando el espacio de memoria disponible es menor que el necesario para almacenar un programa completo se recurre al «overlay»: el programa se divide en segmentos que van entrando, secuencialmente, en la memoria principal.

La planificación de trabajos

La planificación depende del tipo de sistema operativo utilizado, existiendo una gran cantidad de técnicas. La planificación pretende la utilización más eficiente del sistema y lo normal para lograr este objetivo es que trabajos con muy alta prioridad y baja utilización de recursos, se ejecuten antes que otros con baja prioridad pero con mejor utilización de recursos. Las técnicas de planificación más comunes son:

- *Planificación secuencial*

El primer trabajo introducido es el primero en atenderse. Se leen todas las instrucciones y datos de entrada y se almacenan en una memoria de acceso rápido.

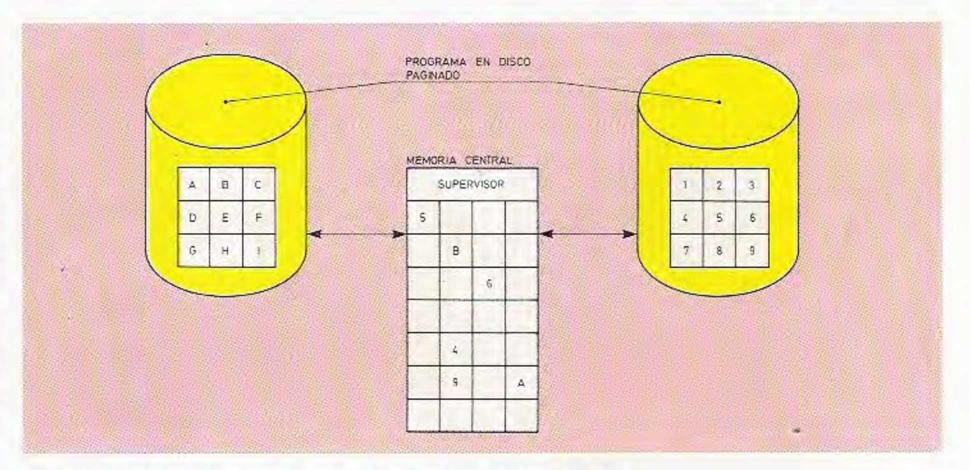
- *Planificación por prioridad*

Esta técnica asigna un código o número a cada programa que indica el orden en el que deben procesarse los trabajos. Los que tienen el mismo número o clase se colocan en colas dentro de la clase. Los trabajos de más alta prioridad son los primeros en ejecutarse.

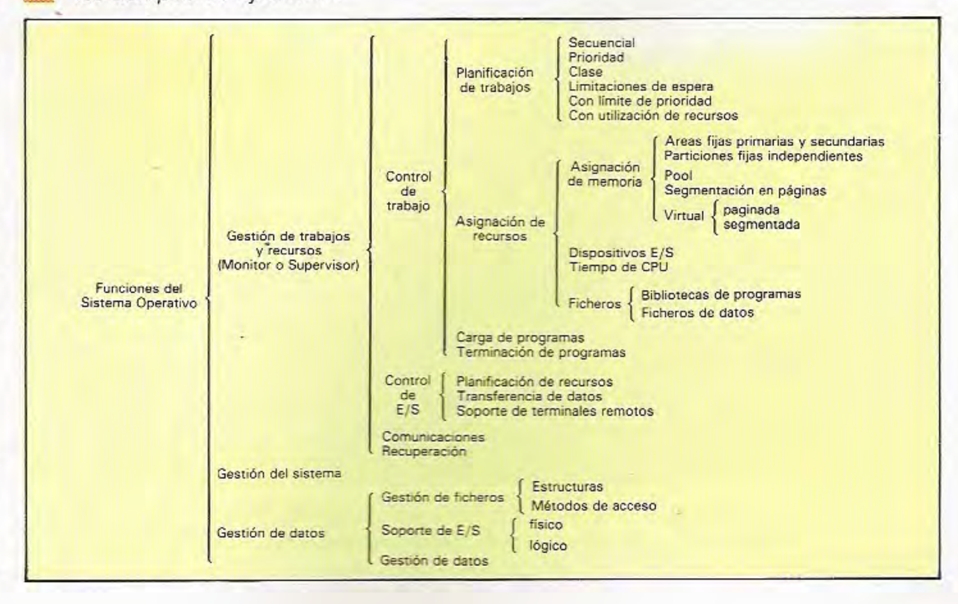
- *Planificación por clase*

Los trabajos se agrupan en clases y en cada clase se atribuye a cada trabajo una prioridad numérica.

Dentro de cada clase se da preferencia a los trabajos que tienen prioridad



Todas las páginas, y por tanto, todas las áreas de la memoria virtual son del mismo tamaño. Esta técnica presenta el inconveniente de aumentar los tiempos de ejecución.



numérica más alta. En general se utiliza en sistemas de particiones de memoria fija: a cada clase corresponde una partición.

- *Planificación con limitación de espera*

Podría suceder, empleando los métodos anteriores, que algunos programas de muy baja prioridad no llegaran a ejecutarse nunca. En este método se asigna un tiempo límite de arranque a cada programa. El sistema operativo puede comprobar si, con la prioridad normal, el trabajo se va a ejecutar o no. En caso negativo se le asigna una prioridad mayor. Si no puede acabar en ese tiempo, el sistema pide la intervención del operador.

- *Planificación con límite de prioridad*

Si en un número determinado de veces el sistema no ejecuta un programa, se le asigna a éste la prioridad más alta con lo que entra rápidamente en servicio.

- *Planificación con utilización de recursos*

En este método el usuario debe estimar los recursos exigidos por el programa, tales como tiempo de CPU, número de líneas de impresión, unidades de cinta magnética, etc. El sistema asigna una prioridad utilizando un algoritmo que pretende optimizar la utilización de los recursos totales.

Asignación de recursos

El sistema tiene que gestionar la asignación de los recursos del ordenador a los distintos programas mediante rutinas particulares que evitan los conflictos entre los diversos programas. El usuario debe indicar, mediante los parámetros del JCL, los recursos que requiere de memoria central, dispositivo de entrada/salida, tiempo de CPU y ficheros.

Muchos sistemas operativos sólo seleccionan un trabajo si todos sus recursos necesarios están disponibles. Esta técnica implica un gran desaprovechamiento del equipo. Por ello es preferible la técnica de *asignación dinámica de recursos,* que permite ocupar los recursos sólo durante el tiempo en que son utilizados.

Otras funciones del monitor

Existen diversas técnicas para la asignación de los dispositivos de E/S en fun-



El sistema operativo simplifica al usuario el empleo de los ordenadores. Consigue, además, racionalizar y optimizar su uso mediante una serie de programas internos entre los que se encuentra la gestión de datos.

ción del tipo de proceso. Pueden ser asignaciones fijas o asignaciones dinámicas.

La asignación de tiempos de trabajo a los distintos programas es realizada por la rutina «dispatcher».

Los ficheros del sistema son de dos tipos: bibliotecas de programas y ficheros de datos.

Los programas y sus rutinas pueden ser exclusivos o compatibles. Las estructuras de programa que manejan más frecuentemente los sistemas operativos son la de recubrimiento, la dinámica y la paginada.

Aunque no todos, muchos sistemas operativos realizan diagnósticos de errores, servicios de temporización, servicios de prueba y depuración.

El *diagnóstico de errores* reconoce tanto los errores de hardware como de software y, una vez solventados, reintenta la ejecución. Si el error permanece pide la intervención del operador. Además lleva un archivo con un histórico de los errores que detecta.

Los *servicios de temporización* permiten parar o arrancar de nuevo un programa al cabo de un cierto tiempo y proporcionar fecha y hora a los programas en ejecución.

Las posibilidades de *prueba y depuración de programas* son muy variadas, permitiendo la corrección de los mismos. *Quizá la posibilidad más interesante* viene dada por el *editor*.

Gestión de los datos

El sistema operativo se ocupa del transporte de los datos entre los dispositivos de E/S y la memoria principal. Para ello es necesario que en la memoria principal se preparen áreas de «buffer».

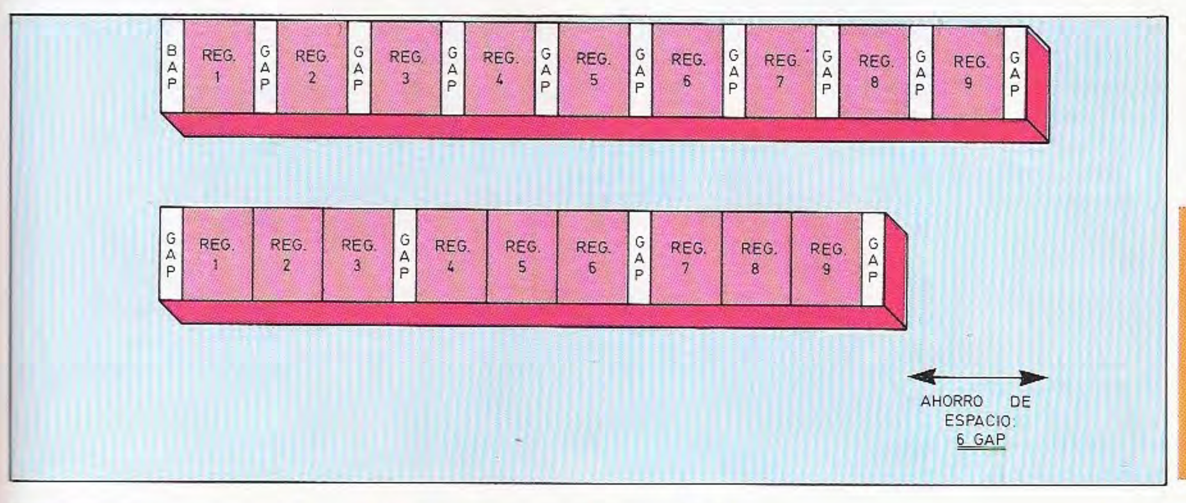
Los programadores de aplicación, sobre todo de los sistemas pequeños, deben reservar parte de la memoria como «buffer» de fichero.

Existen diferentes métodos para la asignación de los «buffer»: buffer único, buffer doble y grupo de buffer.

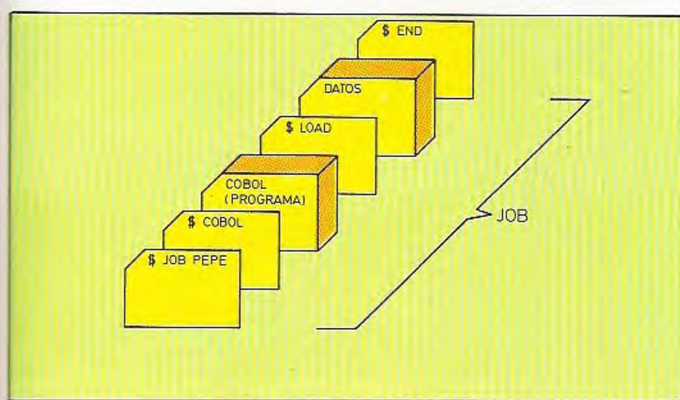
El método de *buffer único* asigna a cada fichero un área fija de memoria independientemente del hecho de que la aplicación esté utilizando o no el fichero.

El método de *buffer doble* es análogo al anterior, pero asigna dos áreas fijas de memoria a cada fichero.

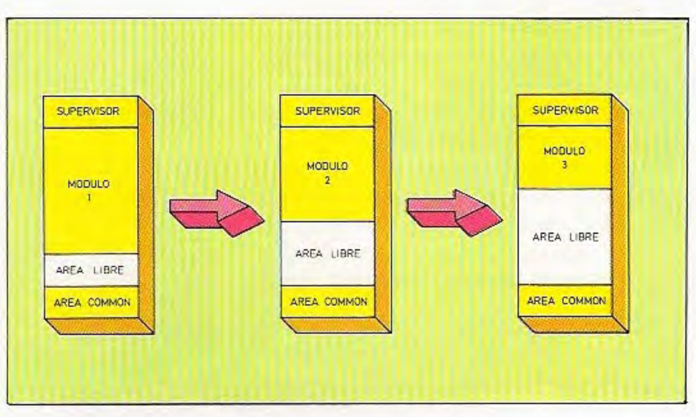
El último método utiliza una amplia porción de memoria, en la que hay varias zonas para los buffers. Cuando es necesaria una operación de E/S en un fi-



El bloqueo de registros en una cinta magnética aumenta la capacidad de almacenamiento de la cinta y disminuye los tiempos totales de acceso a la información.



Ejemplo de un paquete de tarjetas de control que constituye un «job».



Los datos de un programa segmentado se almacenan en un área global («common») lo que permite su empleo en las diversas fases de la ejecución del proceso.

Métodos de asignación de memoria

Cuando se trabaja en multiprogramación hay varios métodos para asignar la memoria:

Asignación de particiones fijas e independientes

Las particiones tienen diferentes dimensiones y a cada programa se le asigna a la partición más pequeña que lo pueda contener.

POOL de memoria

A cada programa se le asigna una cantidad de memoria suficiente para contenerlo. Cuando termina de ejecutarse el programa, la memoria vuelve al pool.

Segmentación en páginas

Las páginas de tamaño fijo y de dimensiones relativamente pequeñas (de 1/2 a 4 K) se mantienen en un pool. Las instrucciones de los programas tienen que dividirse en páginas. Implica el uso de memorias de masa de acceso rápido. El programa reside en memoria secundaria y en la memoria central sólo se encuentran las páginas que están ejecutándose.

Memoria virtual

Se usan discos de gran velocidad para expandir la memoria principal, que de esta forma parece más grande de lo que es realmente. La memoria virtual puede ser paginada o segmentada.
La memoria *virtual paginada* tiene la ventaja de que el programador no tiene que ocuparse de cómo entra el programa en una determinada partición, aunque el sistema operativo aumenta su trabajo de «thrashing».
La memoria *virtual segmentada* utiliza particiones variables, tanto de memoria real como de memoria virtual, basándose en una división lógica del programa en segmentos. Si éstos no son todos iguales hay una infrautilización de la memoria real, ya que hay que reservar memoria para el segmento mayor.

chero, un «buffer» del grupo es asignado al fichero. Cuando acaba la operación, se libera el «buffer» y se devuelve al grupo. De esta forma se consigue un ahorro de la cantidad de memoria correspondiente a los buffers, siempre y cuando no se trabaje con todos los ficheros simultáneamente.

La transcodificación de datos

El sistema operativo llama a las rutinas de transcodificación de datos cuando son necesarias, haciéndolas transparentes para los programas de aplicación.

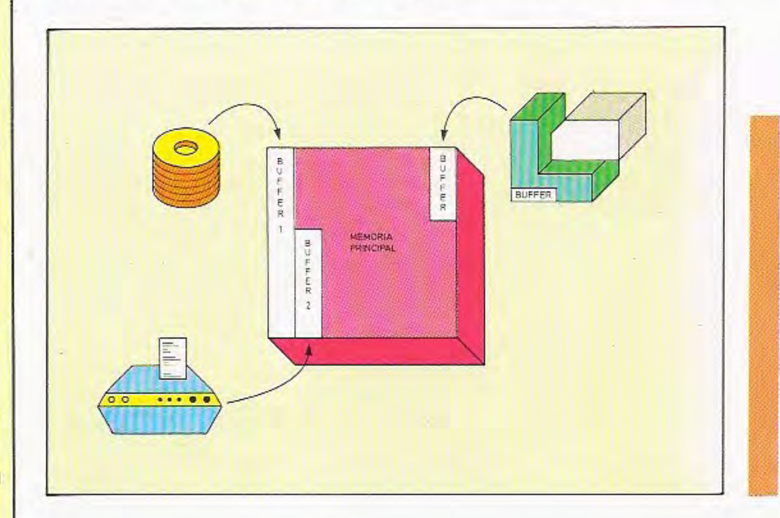
En los sistemas pequeños, muchas veces, la transcodificación tiene que ser resuelta por el programador en su programa de aplicación.

En el caso de entrada de trabajos remotos, el sistema operativo gestiona también las comunicaciones entre los diferentes usuarios.

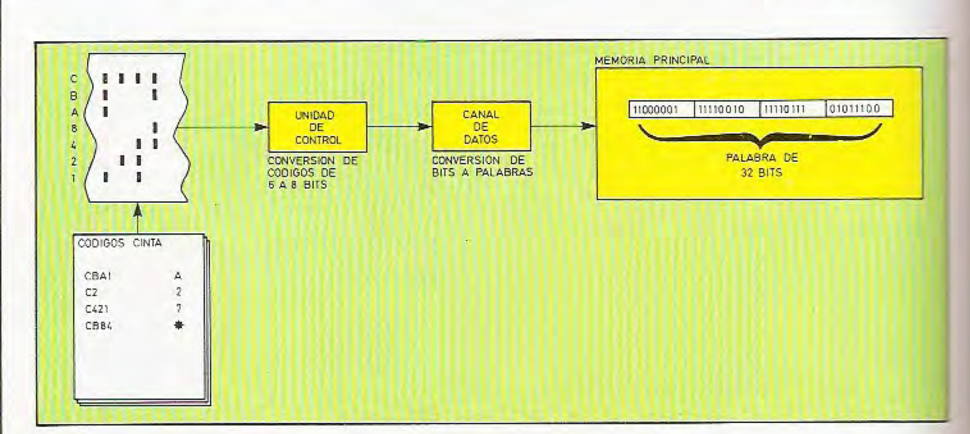
La gestión de datos comprende la gestión de ficheros, el soporte de E/S y del sistema de gestión de datos.

La gestión de ficheros

Las funciones de gestión de ficheros se orientan al control de dichos ficheros. Aunque los ficheros son un conjunto de registros, el sistema de gestión los administra como entidades independientes.



Un buffer es un área de almacenamiento intermedio que libera al procesador de trabajo durante la transmisión de datos. A veces los buffers suelen estar localizados en los mismos periféricos.



Cuando el código empleado para almacenar datos en las memorias de masa difiere del código empleado para almacenarlos en la memoria principal, es necesario recurrir a la transcodificación de datos.

Los ficheros permanentes se identifican con las *etiquetas,* que pueden ser asignadas bien por el usuario o bien por el sistema. La etiqueta de un fichero puede tener diferentes datos, tales como el identificador del fichero, el número de edición, el propietario, la palabra de orden para el acceso, etc.

En los sistemas «batch» y de «time sharing» se mantiene normalmente un catálogo, o directorio, con la localización de todos los ficheros conocidos por el sistema. En el caso de que no se mantenga un directorio, el sistema compara secuencialmente todas las etiquetas hasta que encuentra el fichero que necesita para el programa en proceso.

Muchos de los sistemas operativos incorporan rutinas de utilidad para facilitar las copias de seguridad, de forma que cualquier daño sufrido por los ficheros se pueda subsanar. Estas funciones de recuperación pueden ser iniciadas por el sistema automáticamente, o por el operador, a petición del sistema.

Soporte de E/S

Estas funciones se realizan tanto a nivel físico como a nivel lógico.

Las rutinas que controlan la E/S física llaman a las operaciones de transmisión de datos y gestionan, en parte, el acceso de los programas a los datos, teniendo en cuenta los formatos de transmisión.



Los sistemas operativos se ofrecen como opción al usuario de un sistema informático. Suelen almacenarse en una memoria de masa, generalmente disco magnético, y se cargan en la memoria principal en el instante en que se solicite.

Para saber más

¿Qué es el «overlay»?

«Overlay» significa recubrimiento. Este término designa a la técnica empleada cuando la memoria necesaria para el conjunto de datos y de las instrucciones es mayor que la disponible. La solución consiste en que varios módulos de programa ocupen la misma área en tiempos diferentes, llamando para ello a las rutinas que hagan falta desde las memorias periféricas.

¿Qué es el «overlapping»?

«Overlapping» significa solapamiento. Esta técnica utiliza la transferencia de datos de una parte a otra de la memoria mientras que la ejecución continúa en otro lado.

¿Que es un «Job»?

Un «job» o trabajo es una aplicación global que puede utilizar uno o varios programas de tratamiento, los cuales, a su vez, pueden llamar a otros programas de tratamiento como compiladores, cargadores, etc.

¿Qué es una tarea?

Tarea es la unidad más pequeña de trabajo que puede acceder a los recursos del sistema. Un trabajo se compone, por tanto, de una o varias tareas.

¿Qué es la generación del sistema?

Es la operación que permite adaptar el sistema operativo a la configuración específica del hardware.

¿Qué es JCL?

El JCL es el «Job Control Language», es decir, el lenguaje de control de trabajos. Consta de un conjunto de órdenes que permiten al usuario comunicarse con el sistema operativo.

¿Tiene inconvenientes la paginación?

Sí, ya que el sistema operativo tiene que llevar un mapa de las páginas que están en memoria y en disco. Supone además un fuerte trabajo de «thrashing», es decir, de carga y descarga de páginas entre la memoria central y las periféricas.

En un nivel superior, las rutinas de E/S lógicas permiten la manipulación de los datos con independencia de su estructura física. Estas rutinas constituyen un intermediario entre las operaciones de datos del usuario y la E/S física del sistema.

El soporte de E/S permite a los programas acceder y trabajar con un solo registro del fichero, con lo que el programador no tiene por qué conocer los problemas inherentes a la lectura y escritura de los registros.

Los sistemas operativos gestionan también las estructuras de los ficheros y los métodos de acceso a los mismos. Hay diversas técnicas para asignar las memorias periféricas a los ficheros. En las más sencillas casi toda la gestión, incluida la protección contra destrucciones accidentales, queda en manos de los usuarios; las más complejas asignan dinámicamente los espacios necesarios en los discos cuando son solicitadas.

Las principales estructuras de ficheros son la secuencial, la jerárquica, la de índices, la de listas y la de estructura en bucle.

En la estructura secuencial todos los elementos son del mismo rango y están colocados en serie. En la estructura jerárquica el sistema dispone de un esquema de posición que clasifica y memoriza todos los elementos del fichero. En la estructura de índices el fichero reserva ciertas porciones de memoria para las claves, a fin de localizar la información en el fichero. La característica de la estructura de lista es que cada elemento contiene la dirección del siguiente. En las estructuras en bucle las listas son circulares, es decir, el último elemento de cada una de ellas contiene un puntero con la dirección del primer elemento.

Los métodos de acceso, generalmente soportados por los sistemas operativos, son: acceso secuencial, acceso con índice, acceso con claves y acceso aleatorio.



Muchos ordenadores portátiles basados en el microordenador Z-80 están regidos por el sistema operativo CP/M. El ordenador de la figura utiliza la variante CP/M 2,2.

El acceso secuencial puede realizarse con cualquier tipo de memoria auxiliar y es el único que puede emplearse en cintas y casetes.

La búsqueda de un registro en el acceso con índice se hace a través del directorio.

El acceso con claves es muy útil para unidades de memoria que usan instrucciones de búsqueda cableadas por hardware, ya que de esta forma se libera al procesador de las búsquedas en la memoria secundaria.

Para el acceso aleatorio el sistema utiliza generalmente un algoritmo que establece una correspondencia unívoca entre la clave identificadora del registro y la dirección de memoria en el dispositivo, que necesariamente ha de ser de acceso directo.

Empaquetado y bloqueo

Con el empaquetado se reúne en un único bloque físico a varios registros lógicos. El desempaquetado permite aislar un registro del bloque físico de datos. Los registros pueden ser tanto fijos, como variables. Este método exige un buffer de E/S de mayor tamaño que el registro usado. Esta pérdida de memoria por ocupación de «buffer» queda de sobra compensado por la mejor utilización de la memoria periférica.

El mejor aprovechamiento de las memorias externas obliga también a bloquear los registros. Hay sistemas operativos que sólo gestionan la E/S física, por lo que el usuario tienen que realizar las operaciones de bloqueo y desbloqueo. La gestión de soporte E/S lógico, por los sistemas operativos permite operar a nivel de registro, sin tener en cuenta la estructura de los bloques físicos.

Funciones de manipulación de datos

Las rutinas de manipulación de datos pueden ser llamadas de diferentes formas: por los programas, a través de una



Programa editor

Si durante la compilación o ensamblaje de un programa fuente se detectan errores, es necesario corregir ciertas instrucciones y proceder a un reensamblaje o recompilación. Si el programa fuente se ha introducido utilizando tarjetas es necesario sustituir, añadir o quitar algunas de éstas.

Hoy día es más frecuente introducir los programas directamente en el ordenador mediante un teclado. La necesidad de volver a teclear todo el programa fuente como consecuencia de un error puede ser una gran pérdida de tiempo. Para obviar ese inconveniente existen los programas llamados editores que aceptan órdenes de modificación, adición o supresión de líneas mediante códigos parecidos a los del lenguaje de control.

El texto del programa fuente se almacena en un fichero de disco, y en él se realizan las correcciones.

La operación de edición se realiza tanto en forma interactiva, a través de un terminal de pantalla o un teletipo, o en forma batch, introduciendo las órdenes de edición y los textos corregidos en el flujo de entrada.

La forma más práctica de emplear el programa editor es en modo interactivo. De esta forma se pueden introducir modificaciones en discos que contengan tanto programas fuente, procedimientos compuestos por una secuencia de órdenes de control, como ficheros de datos numéricos o alfanuméricos.

El programa editor evita el uso de un soporte intermedio para la entrada de información al ordenador, a cambio de una entrada más lenta y manual.



Entradas y salidas

Todo ordenador consta de tres partes fundamentales: la CPU, la memoria principal y los periféricos.

La velocidad de trabajo de la unidad central de proceso ha ido aumentando en los últimos años de forma espectacular. Sin embargo, la velocidad de los periféricos no ha aumentado al mismo ritmo. Para que la CPU no tenga que perder tiempo, los periféricos suelen dotarse de procesadores independientes, colocados en una tarjeta de considerable complejidad.

El diálogo entre el procesador principal y los periféricos suele realizarse, casi siempre, durante un proceso de interrupción. En el momento de producirse una interrupción la CPU abandona la tarea que esté ejecutando, guarda las variables de ésta, y salta a una posición de memoria definida por hardware.

En esta dirección el programador habrá escrito una instrucción de salto a la primera posición del programa de control de interrupciones.

Si debe efectuarse un acceso al disco, esta subrutina contiene una orden de lectura que lleva al buffer de su memoria el sector de que se trate; en esta operación están involucrados los bits de sincronismo y los de comprobación interna, entre otros. El sistema proporciona al operador la información almacenada en el buffer correspondiente al registro pedido. Este contenido es colocado en la variable indicada por el programa. Esta tarea es ejecutada por la CPU bajo el control y supervisión del sistema operativo.

tarjeta de control, o mediante la intervención directa del operador.

Estas rutinas son de dos clases: de representación visual y de soporte de periféricos.

Las rutinas de representación visual proporcionan la visualización de la memoria principal, las tablas de los programas, los directorios y los datos que se encuentran en memorias periféricas.

El soporte de periféricos abarca la conversión de los soportes de memoria, edición de datos, enrollamiento de cintas magnéticas, etc.

Aunque no son exclusivas del sistema operativo, las funciones de clasificación y fusión de ficheros están incorporadas a muchos de ellos como utilidades. La fusión de ficheros se realiza sólo cuando éstos se encuentran previamente clasificados.



Enunciados y diagramas de sintaxis

Enunciados de sintaxis

Para definir los elementos de un lenguaje de programación se siguen una serie de convenciones o reglas que varían según las publicaciones. En esta obra utilizaremos indistintamente los *enunciados de sintaxis* y los *diagramas de sintaxis*. Ambos sirven para definir los formatos de las intrucciones de un lenguaje. Las reglas que deben cumplir los enunciados son:

1. Los verbos de las intrucciones y las palabras claves se escriben en MAYUSCULAS. No pueden modificarse al introducirse en el equipo.
2. Las informaciones que proporciona el programador, tales como nombres de variables o número de línea, se escriben en MINUSCULAS.
3. Para indicar la repetición de un término tantas veces como sea necesario, se ponen puntos suspensivos (...).
4. La opcionalidad de una entrada se representa encerrándola entre paréntesis rectangulares ([]). Los paréntesis no se introducen en el equipo.
5. La posibilidad de selección se indica mediante el uso de corchetes ({}) que encierran todas las opciones de las que obligatoriamente debe escogerse una. Los corchetes no se introducen.
6. El uso simultáneo de paréntesis rectangulares y corchetes ([{}]) implica la selección opcional de una de las entradas del grupo.

Los signos de puntuación (,.:;≠, etc.) deben introducirse en el equipo tal como se escriben en las hojas de programación.

Ejemplos:

Sea el formato de una línea de programa

[nnnnn] instrucción [:instrucción...]

como los paréntesis son opcionales, el enunciado de sintaxis indica que son válidos los siguientes formatos:

nnnnn instrucción
nnnnn instrucción-1 : instrucción-2
nnnnn instrucción-1 : instrucción-2 : instrucción-3

y así sucesivamente. También serían válidas las instrucciones

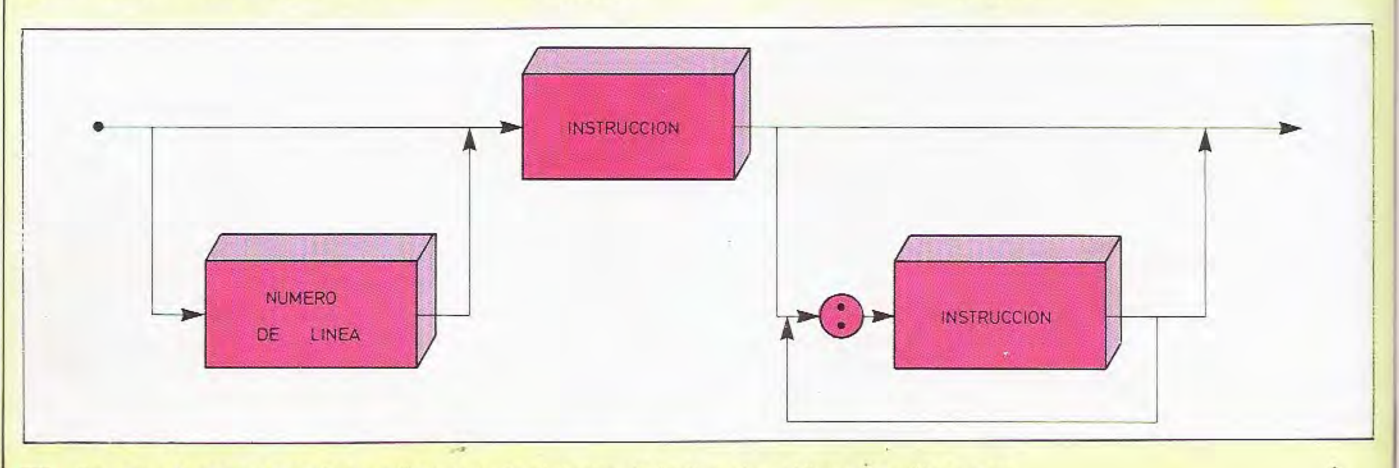
instrucción
instrucción-1 : instrucción-2

Diagramas de sintaxis

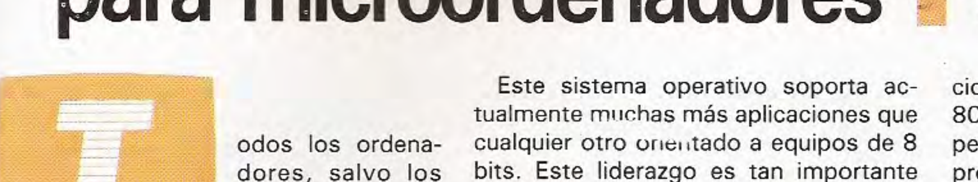
De una forma más gráfica, los diagramas de sintaxis representan la misma información que los enunciados.

Las reglas de uso son:

1. Los verbos de instrucción y las palabras claves se escriben en *Mayúsculas* en el interior de elipses o círculos.
2. Las entradas del programador van en minúsculas, en rectángulos.
3. Los signos de puntuación se escriben dentro de círculos.
4. Los formatos correctos de instrucciones son todas las vías posibles del diagrama.
5. Las entradas situadas por debajo de la línea superior son opcionales.
6. Las entradas en la línea superior son obligatorias.



Ejemplo de estructura de un «diagrama de sintaxis». Este tipo de gráficos se utilizan para representar la estructura sintáctica de los elementos de un lenguaje de programación.



Sistemas operativos para microordenadores

Todos los ordenadores, salvo los más elementales, utilizan sistemas operativos que liberan al usuario de la programación de rutinas de control y le ayudan en el acceso a los soportes magnéticos, listado de ficheros, formateado de soportes, etc.

La menor capacidad y potencia del microordenador no permite el uso de todas las funciones de los sistemas operativos detalladas en los apartados anteriores.

Sistemas operativos para microordenadores

La existencia de equipos basados en microprocesadores de 8, 16 y 32 bits, la posibilidad de ser utilizados por varios usuarios y la cada vez más importante opción de redes de microordenadores, dan lugar a diversos tipos de sistemas operativos.

Así, hay sistemas operativos monousuario para 8 y 16 bits, sistemas operativos multiusuarios para 8 y 16 bits, sistemas operativos concurrentes y sistemas operativos para redes de microordenadores. Aunque algunos microordenadores sólo permiten el uso de un sistema operativo específico, la tendencia actual es que los sistemas operativos sean estándar y puedan, por tanto, utilizarse en diversos equipos.

Sistemas operativos monousuario para 8 bits

Los primeros ordenadores personales de 8 bits, como el Apple II y el Commodore Pet, tenían sus propios sistemas operativos. Más tarde, la compañía Digital Research logró imponer su sistema operativo CP/M como estándar para los microordenadores de 8 bits.

Este sistema operativo soporta actualmente muchas más aplicaciones que cualquier otro orientado a equipos de 8 bits. Este liderazgo es tan importante que la mayoría de los microordenadores de esta categoría permiten a sus usuarios el procesar paquetes de aplicación en CP/M.

El CP/M tenía inicialmente muchos puntos débiles en cuanto a seguridad, recubrimiento de errores y documentación, pero las utilidades que proporcionaba para el manejo de disquetes le proporcionaron rápidamente una fuerte base de aplicaciones.

Las últimas versiones del CP/M han corregido muchos de los fallos iniciales y han incluido el manejo de ficheros, protecciones mediante «password», y gestión de mayor número de unidades de discos y de zonas más amplias de memoria RAM.

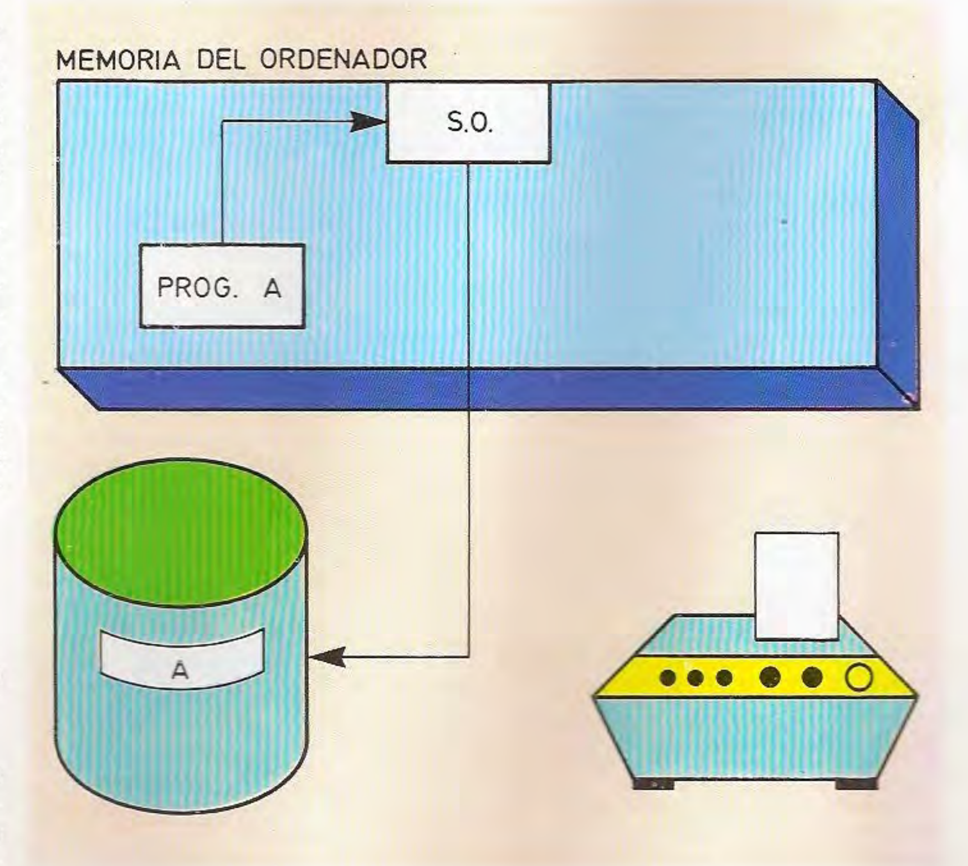
Su principal debilidad es haber sido desarrollado para el conjunto de instrucciones de las familias de procesadores 8080 del Intel y Z80 de Zilog, que han perdido el liderazgo frente a los microprocesadores de 16 bits.

Otros sistemas operativos de este grupo son las versiones para 8 bits del OASIS, UCSD y TURBODOS, que comentaremos más adelante.

Entre el microordenador y la aplicación

Sistemas operativos monousuarios para 16 bits

La aparición de los microprocesadores de 16 bits y la incorporación a los ordenadores personales de discos rígidos han aumentado considerablemente la capacidad de proceso de estos sistemas. El conjunto de instrucciones de los



1. El programa A, que debe imprimir 250 páginas, se inicia. Las páginas se van almacenando en el «spool» en disco.

nuevos microprocesadores, basados principalmente en la familia Intel de 16 bits, difiere significativamente del repertorio propio de los microprocesadores de 8 bits.

Por ello se ha reescrito el CP/M, del que ha surgido el sistema operativo denominado CP/M-86 y, posteriormente, el CP/M Concurrent. Este sistema estaba llamado a ocupar el mismo lugar en el mundo informático que su antecesor. Pero IBM encargó a MicroSoft un sistema operativo para su ordenador personal: el MS-DOS. Como la compatibilidad de software de IBM es una característica importante que intentan ofrecer la mayoría de los fabricantes, el sistema operativo MS-DOS recibió un fuerte impulso, colocándose en cabeza de este grupo de los 16 bits.

El MS-DOS es bastante similar, incluso en el nombre de los comandos, al CP/M-86, con el que además mantiene un cierto grado de compatibilidad. El uso del MS-DOS es más sencillo y el manejo de disco mucho más rápido que con el CP/M-86.

El UCSD es el tercer sistema operativo en cuanto a difusión. Tiene muchas facilidades comunes con el MS-DOS y el CPM-86 y es, además, mucho más compatible y trasladable de un equipo a otro. Sus principales inconvenientes son su lentitud y la rigidez de la estructura de comandos, fuertemente jerarquizada.

La característica principal del UCSD es el uso de un pseudocódigo a nivel de máquina que lo independiza de un microprocesador específico. Con un traductor adecuado, cualquier ordenador puede ejecutar los programas en UCSD. La adaptación de programas requiere sólo la elaboración de los traductores. La lentitud viene provocada precisamente por la traducción del código a lenguaje máquina.

Mediante la utilización del lenguaje UCSD-Pascal desciende la importancia del pseudocódigo, y se incrementa el papel de la compatibilidad entre el sistema operativo y el lenguaje. También soporta FORTRAN 77, Basic, APL y LISP.

Siguiendo el camino del UCSD, la compañía MicroProduts Software ha producido el sistema operativo BOS, para el que existe un lenguaje, el Microcobol, que es un híbrido del PL/1 y el COBOL.

Sistemas operativos multiusuarios para 16 bits

El aumento de capacidad de los ordenadores personales de 16 bits permite el acceso simultáneo a varios procesos, sin tiempos de espera demasiado grandes. Para ello el sistema operativo debe realizar algunas de las funciones típicas de los sistemas operativos de grandes ordenadores, tales como administrar prioridades, encargarse de los protocolos, de los niveles de acceso a los ficheros e interconexión de periféricos. Aunque Digital irrumpió en el área multiusuario con su versión MP/M-86, éste resultó ser un sistema operativo demasiado básico y rápidamente fue desplazado por el UNIX, el cual proporciona un excelente entorno para el desarrollo de programas multiusuario. Las principales facilidades del sistema UNIX son:

- Acceso controlado del usuario.
- Sistema de fichero jerarquizado.
- Lenguaje de comandos seleccionable por la base de usuarios.
- Alto grado de transportabilidad.

Una de sus mejores características es el control de las funciones de escritura y/o lectura a ficheros mediante «password».

El aspecto más importante es su lenguaje de comandos, llamado «shell», tan poderoso que puede ser considerado como un lenguaje de programación. Su potencia viene reforzada por la gran cantidad de rutinas de utilidad suministradas con el sistema. Los comandos del «shell» pueden almacenarse en ficheros, por lo que las tareas («jobs») específicas pueden ejecutarse mediante una sola instrucción. Sus principales debilidades son la falta de retroalimentación interactiva, la pobre consistencia de la sintaxis y de los nombres de los comandos, y el gran número de versiones diferentes que existen.

Dentro de este grupo de sistemas operativos multiusuario para 16 bits destaca el OASIS. Inicialmente fue desarrollado para el microprocesador Z-80, pero ha sido reescrito en lenguaje «C», por lo que es fácilmente incorporable a diferentes hardwares basados en procesadores de 16 bits.

Sus mejores características aparecen en el manejo de ficheros. Estos pueden clasificarse como públicos, privados o de acceso compartido. Los métodos de acceso a los ficheros incluyen las organizaciones directa, secuencial, random y secuencial indexada.

Sistemas operativos para redes de microordenadores

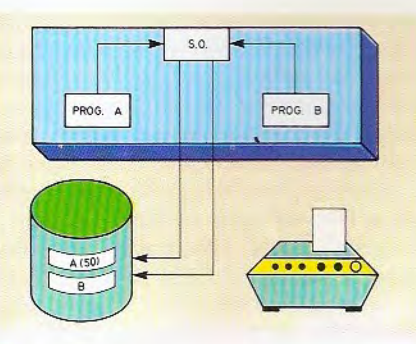
Una alternativa a los microordenadores multiusuario es la interconexión de ordenadores autónomos que permita el acceso compartido a las bases de datos y a los dispositivos periféricos. El funcionamiento de la red se apoya en un sistema operativo adecuado. Uno de ellos es el TURBODOS de Software 2000.

En este sistema un procesador maestro se ocupa del manejo de todos los procesos de discos e impresoras, y los microprocesadores esclavos ejecutan los programas de aplicacion.

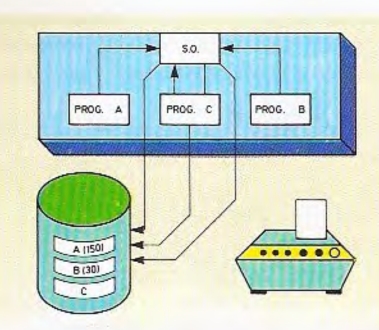
Las principales características del TURBODOS son:

- Cada procesador maestro soporta hasta 16 procesadores esclavos.
- Puede gestionar hasta 16 unidades de discos.
- Spool para 16 impresoras con múltiples colas de espera.
- Compatible con el CP/M2.2.
- Sistemas de correo rudimentario.
- Password de seguridad.

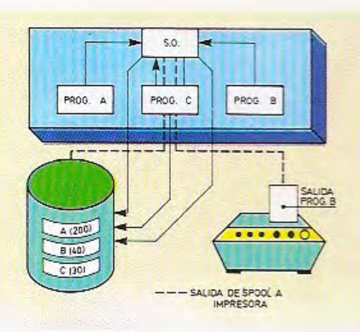
Existe una versión monousuario que ofrece un buen rendimiento en el manejo de discos y en el recubrimiento de errores. Su principal defecto es estar escrito en el código máquina de un microprocesador de 8 bits: el Z-80.



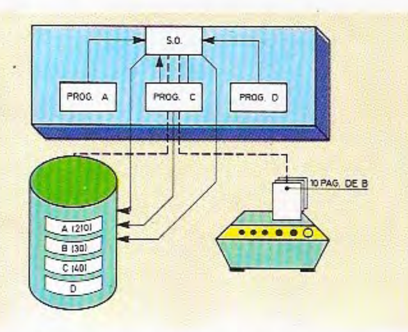
II. *Cuando se han almacenado 50 páginas del programa A empieza la ejecución del programa B y se abre un nuevo fichero para almacenar las 40 páginas que imprimirá este programa.*



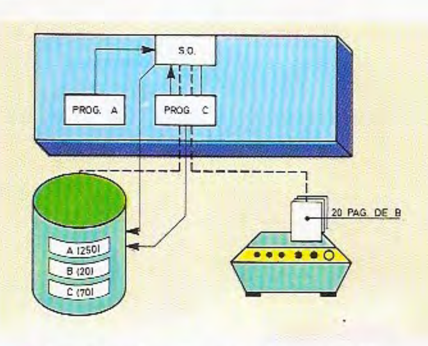
III. *Aún no han terminado de ejecutarse los programas A y B, y empieza la ejecución de un nuevo proceso C. Se abre un nuevo fichero para almacenar sus resultados.*



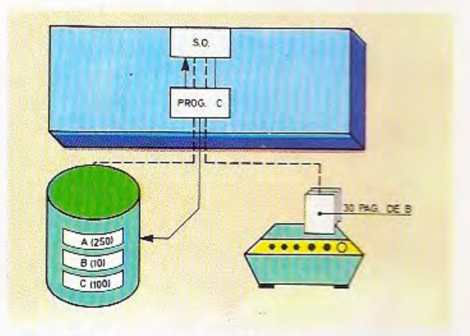
IV. *Finaliza la ejecución del programa B. En el spool hay almacenadas 40 páginas de resultados de este proceso. La impresora, inactiva hasta este momento, empieza a escribir las 40 páginas almacenadas en el spool.*



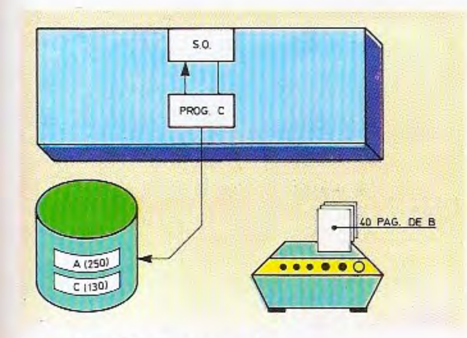
V. *Los programas A y C todavía no han acabado de ejecutarse y del programa B sólo hay 10 páginas impresas. Un cuarto programa D se inicia, para lo que se abre un nuevo fichero en el spool.*



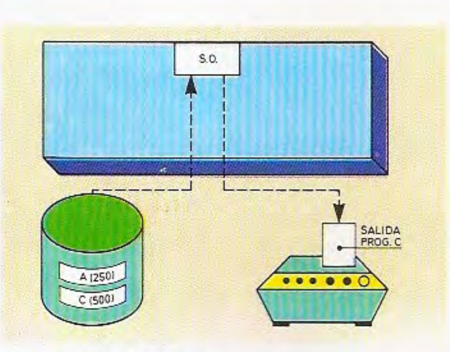
VI. *Al cabo de poco tiempo el programa D «aborta», debido a un error. El operador interviene entonces para anular el archivo D del spool. Ordena, además, retener la salida del A y sacar cinco copias correspondientes al programa C.*



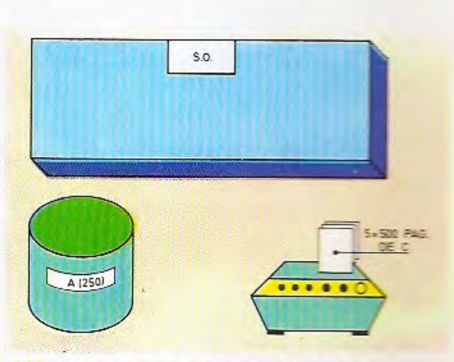
VII. *El programa A finaliza. La impresora está ocupada por el programa B y el C no ha terminado aún de ejecutarse.*



VIII. *El programa B termina su impresión. Como el programa A está retenido, la impresora queda inactiva.*



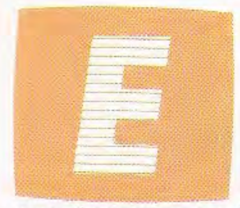
IX *El programa C termina de ejecutarse y comienza la impresión de sus resultados.*



X. *La impresora ha obtenido cinco copias del proceso C. En el spool están almacenados los resultados de salida del programa A, en espera de las instrucciones del operador.*

Introducción al CP/M

El estándar para microordenadores de 8 bits



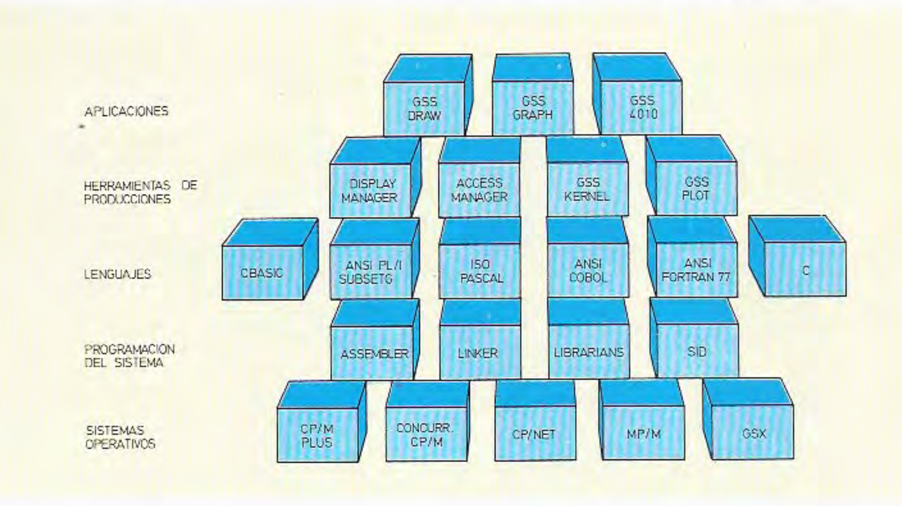
El sistema operativo CP/M nació en 1973, en una época en la que la microinformática era cosa de unos pocos iniciados. La idea de Gary Kildall, su creador, fue desarrollar un sistema que respondiese a estos tres objetivos:

- Que fuese adaptable a cualquier microordenador construido en base a microprocesadores del tipo 8080, 8085 o Z80.
- Que permitiese el almacenamiento de los datos y de los programas del usuario en memorias auxiliares de bajo coste (discos flexibles).
- Que proporcionase un interface software completo y estandarizado para todas las funciones habitualmente requeridas por el programador.

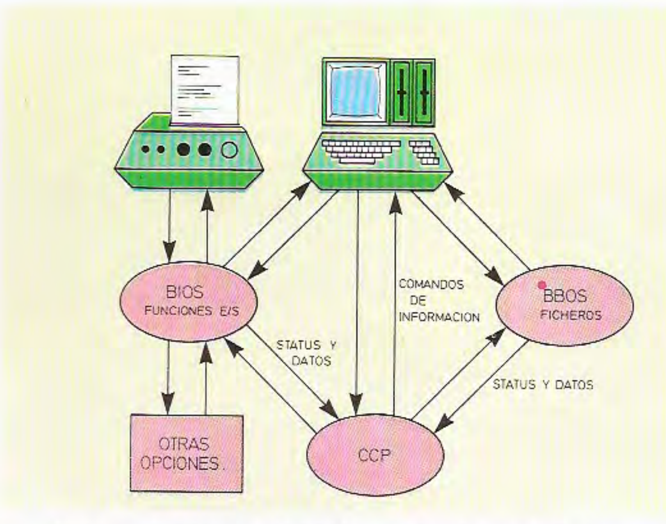
Actualmente el CP/M ha alcanzado tal difusión que se ha convertido en un estándar indiscutible en el campo de los microordenadores basados en un microprocesador de 8 bits. Sin embargo, este sistema operativo no es ni mucho menos perfecto, aunque gusta al gran público y cuenta con el aval de una amplia bilioteca de programas, que van desde lenguajes de programación a programas de aplicación para el tratamiento de textos, la gestión de bases de datos o la comunicación entre ordenadores.

El liderazgo de este sistema operativo en el terreno de la microinformática ha pasado a otras manos con la aparición de los microordenadores de 16 bits equipados con MS-DOS y con el rápido desarrollo de los sistemas multiusuarios/multitarea instruidos por el sistema operativo Unix.

Digital Research Inc, creadora del



Apoyándose en el sistema operativo CP/M se ha construido una gigantesca colección de desarrollos software, que comprende desde lenguajes a aplicaciones.



El sistema operativo CP/M se divide en cuatro partes: BIOS, para el control de entradas y salidas, BDOS para el manejo de ficheros, CCP que sirve de interface entre la consola y el resto del sistema software y, por último, TPA para la conservación de los programas.



El sistema operativo CP/M se ha incorporado a ordenadores de todos los tamaños. El microordenador New Brain, por ejemplo, almacena, en un Kbyte de memoria residente, una variante de este sistema.

CP/M, ha ido lanzando al mercado una extensa variedad de versiones del CP/M original adaptadas a distintos microprocesadores y entornos de proceso. Dentro de la familia destacan el tradicional CP/M (monousuario para equipos de 8 bits), el CONCURRENT CP/M-86 (monousuario para 16 bits), el CONCURRENT CP/M-86, el MP/M II y el MP/M-86 (sistemas multiusuario para 8 y 16 bits), el CP/NET y el MP/NET (sistemas para redes de microordenadores) y el CP/M-68K (para el microprocesador de 16/32 bits 68000 de Motorola), entre otros.

En todo caso, las dos versiones más tradicionales del CP/M son las que se detallan a continuación:

- CP/M-80: sistema operativo para ordenadores monousuario, modular e independiente del hardware. Es utilizable en microordenadores basados en los microprocesadores Intel 8080, 8085 y Zilog Z-80 (todos de 8 bits).

- CP/M-86: orientado a sistemas basados en los microprocesadores de 16 bits Intel 8086, 8088, 80186 y 80286. Permite direccionar hasta 1MB de memoria. Mantiene, por razones de compatibilidad, la estructura de ficheros del CP/M-80.

Estas dos versiones se pueden estudiar simultáneamente puesto que sus estructuras son similares. Entre los lenguajes de programación que pueden ejecutarse en un entorno CP/M se encuentran ADA, ALGOL, APL, BASIC, C, COBOL, FORTH, FORTRAN, LISP, LOGO, PASCAL y PL/1.

El CP/M se divide en cuatro zonas esenciales: BIOS, BDOS, CCP y TPA.

- El BIOS es el programa que define el entorno hardware y que maneja el teclado, pantalla, unidades de disco... Proporciona, en definitiva, los elementos necesarios para la comunicación del ordenador con los periféricos de almacenamiento y comunicación.
- El BDOS se encarga del manejo del disco, controla el directorio de ficheros, permite la ubicación dinámica de espacio y facilita las siguientes operaciones, accesibles todas ellas por el programa:

  - SEARCH: Búsqueda en el directorio del nombre de un fichero.
  - OPEN: Apertura de un fichero.
  - CLOSE: Cierre de un fichero.
  - RENAME: Cambio del nombre de un fichero en el directorio.
  - WRITE: Escritura de un registro en un fichero particular.
  - SELECT: Selección de una sección de disco.

- El CCP sirve de interface entre la consola y el resto del sistema operativo. Lee los datos introducidos por el operador y procesa las instrucciones y controles de operación.
- El TPA conserva los programas cargados bajo el CCP y las correspondientes áreas de datos.

BIOS (FUNCIONES E/S)	BYTE 65935
BDOS (LOCALIZACION ESP. EN DISCO)	
CCP (CONTROL COMUNIC. CONSOLA)	
TPA (PROGRAMAS Y DATOS)	
BUFFER DEL DISCO	BYTE 0

En la figura se representa un mapa de memoria típico de CP/M De los 65.536 bytes ocupados, la mayor parte corresponden al área de programas y datos gestionados por el TPA.

Principales comandos internos

Las principales instrucciones que se utilizan con CP/M son:

- *DIR:* Proporciona un listado con los nombres de todos los ficheros residentes en el disco

- *TYPE:* Lista en pantalla el contenido de un fichero fuente codificado en ASCII y coloca tabuladores cada ocho columnas.
- *REN:* Cambia el nombre de un archivo almacenado en disco y envía un mensaje de respuesta «FILE EXISTS» o «NO FILE», para indicar si el fichero renombrado existe o no.
- *ERA:* Borra un archivo del disco. Existe un formato especial para borrar todos los ficheros de un disco.
- *SAVE:* Almacena el contenido de toda la memoria del sistema en disco.
- *USER:* Define áreas de usuarios en el directorio.

Comandos transitorios

Los comandos transitorios son llamadas a programas específicos. Algunos comandos básicos son suministrados por el sistema y otros comandos son definidos por el usuario. Estos comandos son cargados desde el disco y se ejecutan en el TPA. Entre ellos destaca el comando PIP.

El PIP (Peripheral Interchange Program o Programa de intercambio con periféricos) realiza las operaciones básicas de carga, impresión, copia y combinación de ficheros de datos. En el formato del comando se indica el periférico o fichero que recibe los datos y los ficheros o dispositivos que contienen la información a copiar. Las instrucciones pueden incorporar diversos nombres como NUL (envía 40 ceros ASCII), EOF (envía un fin de fichero), etc. y parámetros adicionales entre corchetes para indicar opciones, como H (transferencia de datos hexadecimales), TN (insertar tabuladores cada n columnas), B (transferencia en bloques), etc.

Existen otros comandos transitorios como:

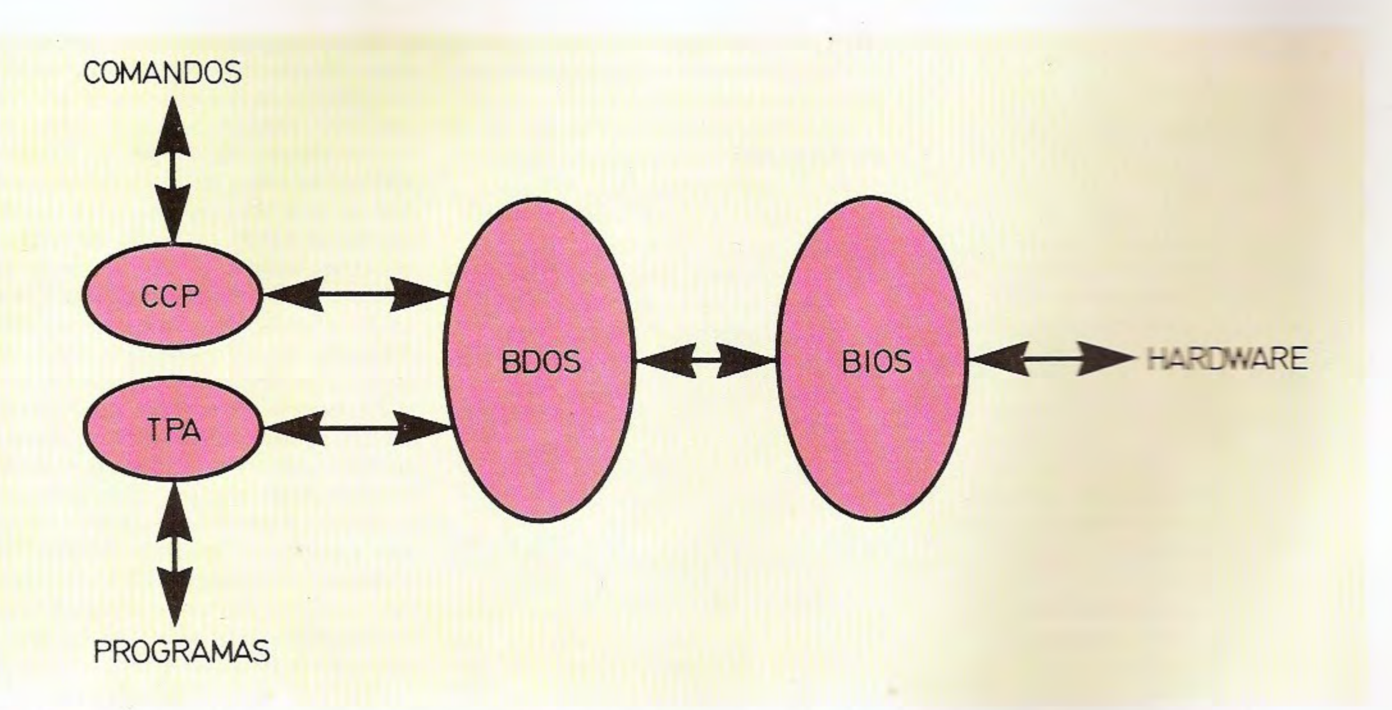
- *STAT:* Proporciona información estadística sobre el espacio del disco, los ficheros y la asignación de dispositivos.
- *LOAD:* Carga ficheros hexadecimales y produce ficheros de programas ejecutables.
- *DDT* (Dynamic Debugging Tool): Este comando es una herramienta de depuración.
- *SYSGEN:* Genera el sistema operativo en función de la configuración del equipo.
- *SUBMIT:* Carga un fichero de comandos para procesos batch.
- *DUMP:* Vuelca el contenido de un fichero en hexadecimal.
- *CONFIG:* Cambia los parámetros de configuración.
- *FORMAT:* Prepara o formatea un disco.

El programa editor

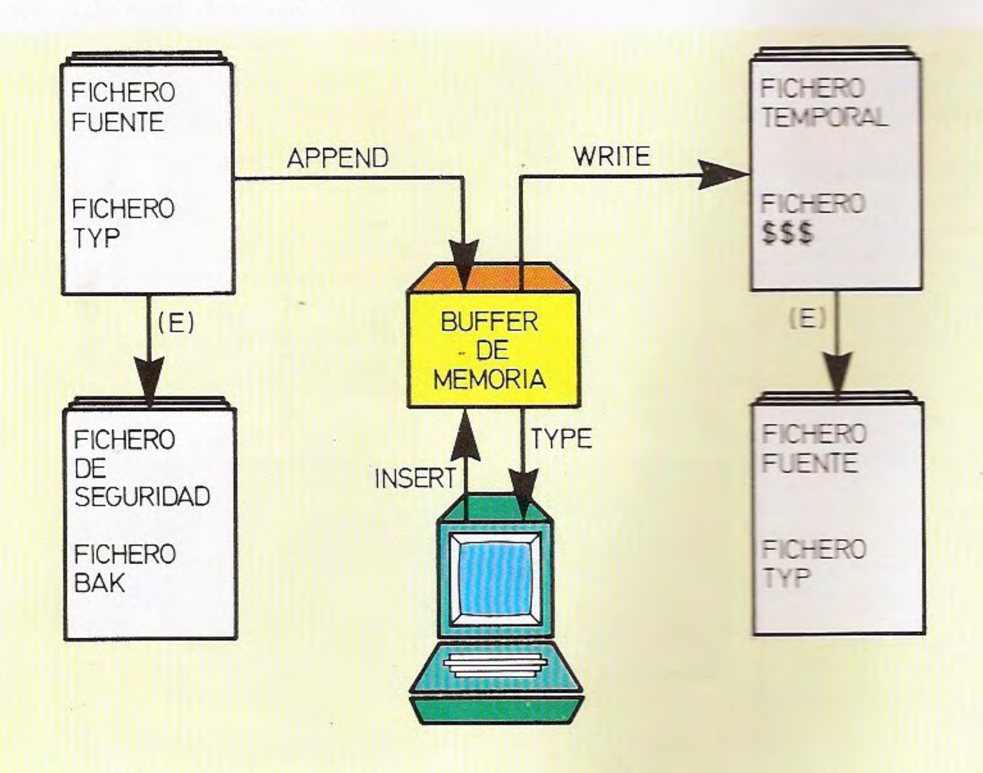
El comando ED llama a un programa que permite la creación y modificación de ficheros ASCII. Estos ficheros están organizados como una secuencia de caracteres separados por caracteres de fin de línea. La longitud de ésta no está sujeta a ninguna restricción. Este programa resulta muy útil para corregir programas almacenados en ficheros, ya que sus comandos permiten la búsqueda, la sustitución o la inserción de cadenas de caracteres. Facilita también la creación y apertura de nuevos ficheros.



La versión CP/M 86 es empleada por ordenadores de 16 bits. Este ordenador incorpora el microprocesador 8088 y se gestiona mediante esta variante de CP/M.



Estructura funcional del CP/M. El módulo BIOS sirve de interface entre el hardware y el resto del sistema. El BDOS gestiona la memoria; el CCP, por último, contiene los comandos y se encarga de controlar la consola.



Ficheros manipulados con CP/M. Por medio del editor, el usuario puede llevar parte, o la totalidad de un archivo ya creado a la memoria del sistema, y listarlo, o modificarlo después, como quiera.

La historia del CP/M

En 1973, el joven Gary Kildall prestaba sus servicios en Intel como encargado del área de lenguajes para el microprocesador 8080.
Cuando necesitó trabajar con un disco flexible, concretamente con un prototipo de Shugart, desarrolló un programa que permitía controlar algunas interacciones entre el microprocesador y el disquete. Había nacido el primer sistema operativo para microordenadores de 8 bits.
Sin embargo, el proyecto no fue desarrollado por la compañía Intel, que rechazó la oferta de Kildall para continuar con él y adoptó otro sistema operativo.
Kildall y su esposa Dorothy decidieron introducir en el mercado el producto obtenido, al que denominaron «Control Program for Microprocessors» (CP/M). Pensaban en la multitud de personas que ya empezaban a construir pequeños ordenadores personales. La compañía que Gary y Dorothy crearon, la Intergalactic Digital Research, nació y se estableció en el cuarto de juegos de su casa.
En los primeros tiempos, Dorothy llevaba el negocio mientras que Gary enseñaba informática y cálculo numérico en la cercana Escuela Naval de Monterrey.
Este sistema operativo al principio era sólo un producto para expertos, apenas conocido por los lectores de revistas para aficionados. Su bajo precio (unos 70 dólares) y su particularidad de ser el único sistema operativo que controlaba unidades de discos flexibles hizo que el negocio fuera incrementándose, por lo que Kildall dejó sus clases y se dedicó de pleno a su empresa familiar.
El desarrollo espectacular del CP/M empezó en 1976, cuando todavía existían pocos fabricantes de ordenadores personales.

ED crea un fichero de trabajo intermedio que almacena los datos editados durante el proceso.

Las funciones y comandos del ED se clasifican en:

- Funciones de transferencia de texto.
- Numeración de líneas.
- Operaciones de Buffer de Memoria (inserciones de líneas).
- Cadenas de comandos.
- Búsqueda y alteración de textos.
- Bibliotecas fuente.
- Ejecución repetitiva de comandos.

Hardware, necesario para el CP/M

La concepción modular del CP/M permite su incorporación a máquinas construidas sobre cualquier microprocesador que contenga el juego de instrucciones del 8080, como es el caso del 8085 y del Z-80. Esto es debido a que los módulos básicos del CP/M (CCP, BDOS, BIOS), han sido escritos en el ensamblador del 8080. Es por ello que el sistema operativo CP/M no es transportable a cualquier ordenador, como ocurre, teóricamente al menos, con los sistemas UCSD y UNIX, que están escritos en los lenguajes de alto nivel PASCAL y C, respectivamente.

El sistema no ocupa más que 6,5 Kbytes de memoria RAM, aunque un cierto número de productos estandarizados (editor de textos ED, ensamblador ASM, etc.) que funcionan bajo CP/M y son suministrados con el sistema necesitan un mínimo de 16 Kbytes. El tamaño medio de la memoria ocupado por la configuración máxima alcanza los 64 Kbytes.

Para que un ordenador funcione con

	MONOUSUARIO MONOTAREA	MONOUSUARIO MULTITAREAS	MULTIUSUARIOS MULTITAREAS	RED DE ORDENADORES
8 BITS	CP/M	MP/M	MP/M MP/M II	CP/NET MP/NET
16 BITS	CP/M-86	MP/M-86 CCP/M-86	MP/M-86	CP/NET-86 NP/NET-86

La gama de sistemas operativos para microordenadores desarrollados por los creadores del CP/M, se han ido ampliando, hasta cubrir, hoy día, todas las posibles necesidades de los sistemas pequeños.



Ningún fabricante de ordenador se atreve a lanzar un nuevo producto al mercado sin tener en cuenta la gran variedad de productos software desarrollados para CP/M.

CP/M debe disponer de un soporte magnético directamente direccionable, esto es, de una o varias unidades de disco.

Descripción funcional del CP/M

En toda máquina en la que se implante este sistema debe existir un cargador en «frío» del sistema implementado en una memoria ROM. Este cargador tiene encomendada la misión de arrancar el sistema y cargar en la memoria RAM los tres módulos que componen el CP/M (CCP, BDOS y BIOS).

El módulo CCP es, esencialmente, un intérprete de comandos. Lee de la consola las órdenes tecleadas por el usuario y las analiza sintácticamente antes de proceder a su ejecución.

Una vez que la sintaxis del comando ha sido aceptada, el CCP lo carga en un archivo de tipo COM, emplazado en una zona de la memoria reservada para el programa de usuario, la zona TPA.

La solicitud de comando se señala mediante la aparición en el terminal de un «prefijo» o «prompt» de la siguiente forma: A>

En donde A es el nombre del disco que se está utilizando.

Existen dos categorías de comandos controlados por CP/M: los comandos residentes, integrados en el sistema



El sistema operativo MP/M es un sistema operativo compatible con el CP/M para microordenadores de 8 bits que operen en multitarea. La CPU del ordenador de la figura es un Z-80, lo que permite operar con este sistema operativo.

MICRO-PROCESAD.	MONOUSUARIO		MULTIUSUARIO	
	MONOTAREA	MULTITAREA	MONOTAREA	MULTITAREA
8080 Z-80	CP/M PLUS	MP/M II	MP/M II	CP/NET
8086	CP/M-86	CP/M CONCURR.	MP/M-86	CP/NET 86
6800	CP/M 68K			
Z8000	CP/M 78K			

La familia de sistemas operativos CP/M puede trabajar con distintos tipos de microprocesadores de 8 ó 16 bits, con un sistema monousuario o con otro multiusuario. El gráfico representa las principales posibilidades de este sistema.

Para saber más

¿El sistema operativo CP/M puede incorporarse a cualquier microordenador?

No, sólo puede incorporarse a sistemas basados en los microprocesadores que contengan el juego de instrucciones propio del 8080; tal es el caso de los microprocesadores 8085 y Z-80.

¿Además del tipo de microprocesador, el CP/M plantea otras exigencias hardware para su incorporación a un ordenador?

Para que sea posible la adaptación del CP/M es preciso que el ordenador posea el suficiente espacio de memoria y disponga de al menos un soporte magnético directamente direccionable.

¿Puede utilizarse el sistema operativo CP/M en ordenadores de tipo multiusuario?

Este sistema operativo está desarrollado exclusivamente para funcionamiento monousuario, no obstante, existen derivados del CP/M convencional diseñados para ordenadores multiusuario (MP/M).

¿Cuál es la estructura de la memoria controlada por el sistema operativo CP/M?

El CP/M precisa de cinco zonas de memoria para su implementación y funcionamiento. Las tres primeras, destinadas al almacenamiento de los módulos constitutivos del CP/M (CCP, BDOS y BIOS). Las dos zonas restantes están destinadas a los programas del usuario (TPA) y al almacenamiento de parámetros y vectores utilizados por el sistema operativo (SPA).

operativo dentro del módulo CCP, y los no residentes, grabados en disco en forma de archivos estandarizados de tipo COM, lo que les confiere la propiedad de ser archivos ejecutables.

Cada disco controlado por CP/M contiene un catálogo o «directorio» propio. Este catálogo permite obtener un repertorio de los archivos presentes en el disco.

En la versión 1.4 del sistema operativo CP/M no existe el concepto de propietario. Cualquier usuario puede acceder a todos los ficheros presentes en el sistema. Los discos, en la versión CP/M 2.2, están divididos en varias zonas lógicas, cada una de las cuales puede asignarse a un usuario diferente.

El modulo BIOS

El BIOS contiene el conjunto de programas que gestionan las entradas y salidas, programas que son específicos de la configuración hardware adoptada.

El conjunto de estos programas externos está generalmente ideado y escrito por el fabricante del microordenador. El programa fuente del BIOS, escrito en lenguaje máquina, ensamblador o macroensamblador, se suministra al usuario con el resto del sistema operativo, contemplando la posibilidad de que éste quiera sustraer o añadir programas específicos, en función de sus necesidades propias (por ejemplo, para el control de periféricos de entrada o salida no estandarizados).

La conexión o «interface» entre el BIOS y el resto del sistema se realiza a través del módulo BDOS.

Estructura de la memoria

La memoria controlada por el sistema operativo CP/M se divide en cinco zonas. El cargador específico del sistema implanta los tres módulos del CP/M en la parte alta de la memoria. La parte baja se divide en dos zonas: la TPA, reservada para los programas del usuario, y la SPA donde el sistema operativo guarda algunos parámetros y vectores de salto a subrutinas.

Los comandos del CP/M

Para introducir un comando debe aparecer en pantalla el «prompt» de aviso, de la forma A>. Existen dos tipos de comandos: los integrados en el sistema operativo CP/M, también denominados residentes, y los comandos externos suministrados con el sistema y asociados a archivos ejecutables. Estos archivos suelen estar localizados en el disco que contiene el sistema operativo.

Los comandos residentes son comandos de interés general y de uso muy frecuente. Como ocupan muy poco espacio están colocados junto al CCP, en la memoria central. Entre ellos se encuentran el comando DIR, que proporciona un listado de todos los archivos presentes en el disco seleccionado; ERA, con el que se ordena el borrado de uno o varios de los archivos de un disco, y SAVE, que guarda en el disco todos los programas residentes en la zona TPA.

En el disco de sistema operativo suelen viajar también algunos ficheros correspondientes a comandos externos o no residentes.

Algunos de estos comandos poseen subcomandos propios. Se accede entonces a auténticas utilidades que gestionan un diálogo con el usuario; cabe citar, por ejemplo, los comandos de utilidad ED y DDT.

Servicios del sistema

El CP/M facilita dos tipos de servicios: las funciones del BDOS y las primitivas del BIOS.

Las funciones lógicas ofrecidas por el módulo BDOS son accesibles a cualquier programador. Las primitivas del BIOS (funciones modificables de entrada/salida) dependen del entorno hardware en el que se trabaje. El empleo de estas funciones primitivas resulta algo complicado.

El acceso a las funciones del sistema se realiza introduciendo el número de la función deseada o, en algunos casos, mediante la instrucción de llamada CALL.

Estas funciones permiten operar sobre el terminal y la impresora (lectura o escritura de caracteres o líneas, por ejemplo); con los discos (selección de uno de ellos, protección, lectura del vector de protección, etc.); operar sobre los archivos (abrir, cerrar, buscar, alterar el nombre, leer, escribir, posicionarse sobre un determinado registro...); entre otros cometidos.

El sistema operativo CP/M concurrente

El CP/M Concurrente es un sistema operativo monousuario/multitarea para microcomputadores basados en el 8086 y 8088. Es compatible con los sistemas CP/M-86 y MP/M-86. Los comandos tradicionales del CP/M de 8 ó 16 bits están presentes en el concurrente y tienen su misma sintaxis; si bien, incorpora además nuevas instrucciones, como DSKMAINT (formateado y copia de disquetes) o SHOW (visualización de las características de un disquete).

En este sistema operativo aparece el concepto de consola virtual, que ofrece el equivalente a cuatro puestos de operadores simultáneos y visualizables sobre una única consola física.

El CP/M concurrente utiliza, además de una metodología de tiempo compartido, una potencialidad de tiempo real, por lo que el sistema tiene una operatividad inmediata a requerimientos externos. Esta particularidad lo hace especialmente útil en procesos técnicos, científicos e industriales.

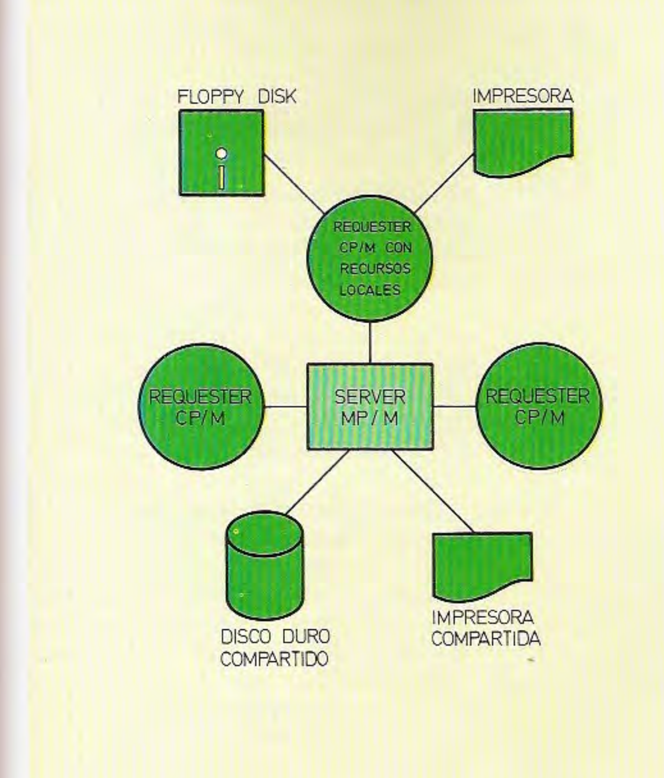
El núcleo, llamado RTM (tiempo real del monitor) desempeña las funciones de *Process Dispatching, Polling,* gestión de subrutinas, flags y temporización del sistema.

El *dispatcher* utiliza dos áreas, llamadas PD y UDA, para leer y memorizar el estado de un proceso activo. Estos estados pueden ser tres: preparado (ready), activo (running) e interrumpido (suspended). Otro elemento del núcleo es el gestor de subrutinas (Queue Management), que trabaja principalmente con los procesos interrumpidos y en el que el Mutual Exclusion Queue funciona como «semáforo» garantizando que en un momento determinado sólo un proceso accede a una fuente del sistema.

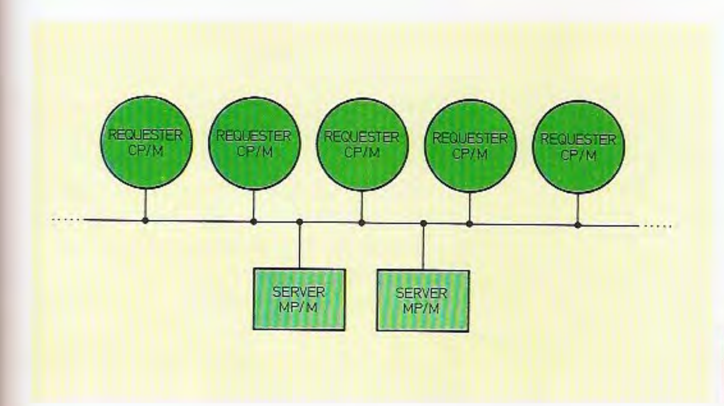
Por último, el temporizador se apoya en un proceso interno de reloj que tiene constantemente el horario real. El sistema puede poner en marcha una función de retraso (Delay) para programar una ejecución diferida en el tiempo.

Los otros módulos del CP/M Concurrente son: MEM (Memory Management Module) para la gestión de memoria; CIO (Character Input/Out Module), que gestiona la entrada/salida sobre el monitor y la impresora; SCREEN (Virtual Console Screen Manager), para la gestión del monitor, y BDOS (Basic Disk Operating System).

El BDOS del CP/M Concurrente es una ampliación del BDOS del CP/M-86. Incorpora elementos de protección para la



Estructura en estrella del CP/NET. Permite una configuración dotada tanto de requester con periféricos locales (CP/NET), como otra compuesta de sólo la unidad central y la consola (CP/NOS).



Configuración del CP/NET de canal común.

operación multipuesto y tiene una gestión de archivos mucho más sofisticada.

Sistema operativo MP/M

EL MP/M (Multi-Programming Monitor) es un sistema operativo en tiempo real multiusuario y multitarea que tiene dos versiones:

- El MP/M para microprocesadores 8080, 8085 y Z-80.
- El MP/M-86 para los 8086 y 8088.

Este sistema es muy similar al CP/M Concurrente. Se diferencia básicamente en la gestión de la consola: es capaz de gestionar físicamente consolas distintas. Se basa en un núcleo de tiempo real con mecanismo de prioridad de 256 niveles y con rotación entre procesos, por lo que es también un sistema de tiempo compartido.

El *dispatcher,* así como los mecanismos de *Mutual Exclusión* y las funciones de temporización y manejo de archivos con sus elementos de seguridad, son análogos al del CP/M Concurrente. Además de la asociación de un proceso a un terminal físico, el MP/M permite manejar un número mayor de procesos con un solo terminal.

Sistema operativo CP/NET

El CP/NET actúa como puente entre procesadores host que manejan los recursos compartidos bajo MP/M y ordenadores terminales que desarrollan funciones de puestos de trabajo bajo CP/M. Los primeros desarrollan funciones *server* o de aprovisionamiento, y los segundos de *requester* o solicitud.

Cada solicitante puede operar sólo bajo CP/M con sus propios recursos de periferia y memoria sin tener que recurrir a un aprovisionador.

Una variante de este sistema es el CP/NOS, que puede ser utilizado como solicitador cuando no se dispone de memoria de masa propia. La red, en este caso, comparte la memoria del aprovisionador.

Los principales módulos del CP/NET son:

- *NDOS:* Extensión del BDOS, parte del sistema operativo almacenado en disco.
- *SNIOS* (Slave Network I/O System): Adición al BIOS. Trabaja como nexo de unión del NDOS y una unidad física particular de la red. Sirve para configurar el sistema según la estructura y topología deseada.
- *CCP* (Console Command Processor): Elemento que sustituye al CCP regulador del CP/M.
- *NETWRKIF* (Network Interface Process): Engloba las extensiones del MP/M para la gestión física de los *Servers.*

Las siete funciones del SNIOS son:

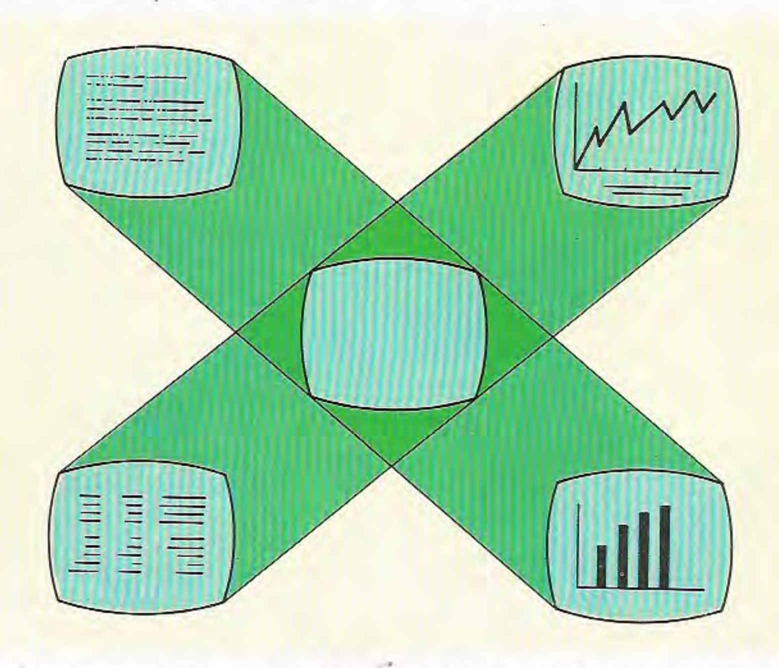
- Envíos de mensaje a la red.
- Recepción de mensajes desde la red.
- Reporte de errores de red.
- Proceso «Warm boot» de la red.
- Devolución del estatus de la red.
- Devolución de la configuración de la tabla de direcciones.



CP/M Plus

Los microprocesadores de ocho bits pueden manejar hoy día más de 64 Kbytes de memoria central gracias a mecanismos hardware/software de «memory management». Por ello, Digital ha potenciado los sistemas de 8 bits mediante el CP/M Plus.

Este sistema es un desarrollo del CP/M 2.2 con una mayor funcionalidad y potencia. Puede manejar hasta un Megabyte de memoria central, unidades de disco de 512 Mbytes o programas de I/O sobre sectores múltiples. El sistema se presenta en dos versiones: la almacenada, para sistemas de memoria extendida y la no almacenada para los sistemas existentes de sólo 64 Kbytes de memoria RAM. Esta última versión no utiliza todas las funciones avanzadas. El CP/M mantiene en cada banco de memoria una parte residente del BDOS (1,5 Kbyte) y del BIOS (algo menos de 2 Kbytes) lo que mantiene la integridad operativa del sistema cuando se traslada el control de un banco de memoria a otro.



El CP/M concurrente incorpora la posibilidad de operar simultáneamente con cuatro pantallas virtuales. Estas son visualizadas inmediatamente ante cualquier consulta física.

Introducción al OASIS

Un sistema operativo mono/multiusuario para 8 y 16 bits



El sistema operativo OASIS está diseñado para trabajar en un entorno tanto monousuario como multiusuario. La gama de periféricos que puede controlar es muy amplia: impresoras, terminales, discos, etc. Está diseñado en su origen sobre el microprocesador Z-80.

Objetivos de diseño

En el diseño de este sistema operativo se intentaron alcanzar los siguientes objetivos:

- Desarrollar un sistema capaz de implementarse en ordenadores pequeños, pero que contara con características propias de los sistemas operativos para equipos de tamaño medio.
- Implantar este sistema en una gran variedad de máquinas construidas por diferentes fabricantes.
- Crear un sistema operativo destinado a usuarios no especializados en informática.
- Ofrecer un sistema operativo que permitiese un fácil diseño de software.

La consecución del primer objetivo resultó sencilla: programadores de sistemas mayores ofrecieron su experiencia en el diseño del OASIS.

La mayor dificultad residió en conseguir que fuese compatible con máquinas de distintos fabricantes. Esta característica presupone la independencia del sistema operativo respecto al hardware de la máquina con la que trabaje. Esto se consiguió en parte mediante un diseño modular que permite introducir cambios de forma sencilla en cualquiera de los módulos.

La sencillez de manejo para el personal no especializado en informática se consigue mediante la incorporación de comandos, cada uno de los cuales contiene «ayudas» específicas que facilitan la formulación de las instrucciones adecuadas.

Mediante el diseño de funciones intrínsecas del propio sistema se facilita el posterior desarrollo de aplicaciones y programas.

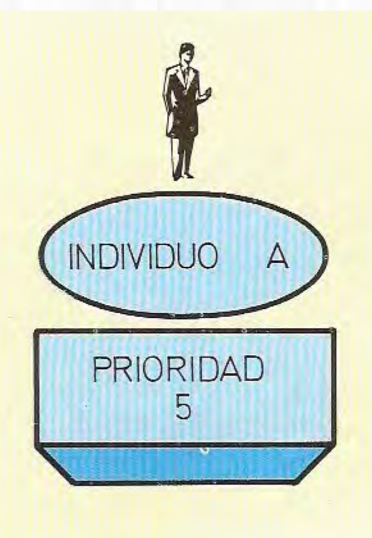
Partes del sistema

El sistema operativo OASIS se compone, en sus primeras versiones, de una serie de programas agrupados en los siguientes bloques:

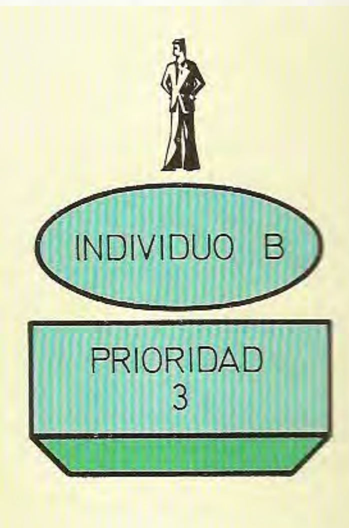
- *Núcleo* (System Nucleus): Comprende un conjunto de subrutinas genéricas que se cargan en el ordenador al hacer el «bootstrap». Si se está trabajando en régimen multiusuario el núcleo es quien gestiona los recursos compartidos. En definitiva, este conjunto de programas es el encargado de controlar a todos los dispositivos conectados al sistema.
- *Command String Interpreter* (CSI): Es un intérprete que permite acceder al sistema y a los programas de usuario, indistintamente. Se encarga también de la comunicación entre los distintos módulos que componen el sistema completo.

El desarrollo de software bajo el sistema operativo OASIS se realiza de forma sencilla gracias a una serie de ayudas. Entre ellas se encuentra un editor que permite la creación de textos.

Mediante este editor se accede al *Macro Assembler* incorporado, que es quien llama a las funciones propias del sistema y a las directivas. Este sistema operativo está dotado, igualmente, de un montador de programas para la reubicación del código objeto. Un programa depurador de errores permite el cambio de registros, la inclusión de puntos de rup-







El sistema operativo OASIS asigna, a cada usuario, distintas prioridades de acceso a los ficheros. En el ejemplo de la figura el individuo B puede consultar todos los registros con una prioridad menor o igual que tres, mientras que el individuo B tiene un acceso mucho más restringido.

tura y el desensamblaje de instrucciones. Por último, el OASIS se acompaña de un intérprete/compilador de Basic.

Tratamiento de ficheros

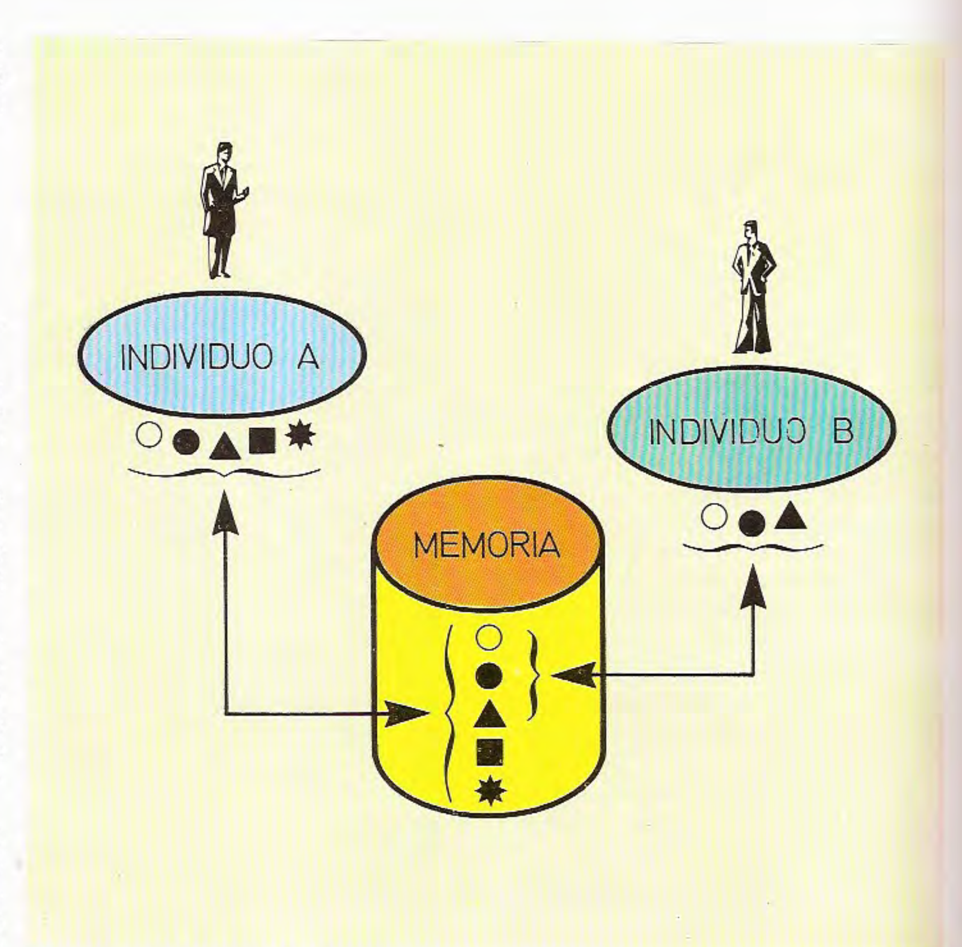
Este sistema operativo permite el acceso restringido a ficheros.

Cada usuario puede crear ficheros privados, de forma que sólo él pueda efectuar consultas, modificaciones o borrados.

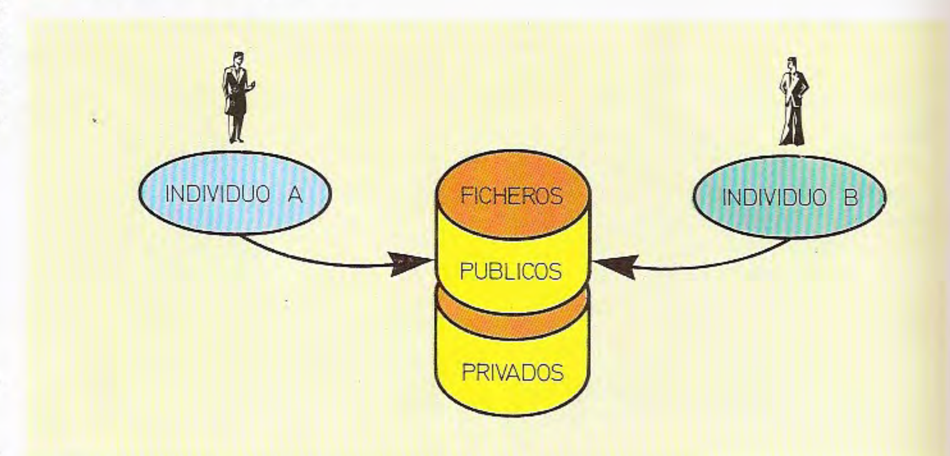
Dispone, igualmente, de la opción de ficheros compartidos en los cuales varios usuarios pueden efectuar consultas y modificaciones, aunque no les está permitido su borrado.

Un tipo especial de ficheros compartidos son los ficheros públicos cuyo propietario es el propio sistema operativo y a los que pueden acceder todos los usuarios del sistema. Ficheros de este tipo son los intérpretes y los compiladores.

Otra de las opciones del sistema es el acceso restringido a algunos niveles de comandos. En el momento de la creación de uno de ellos se define el nivel de privilegio que puede tener el usuario. El nivel 1, por ejemplo, permite «ver» todos los ficheros existentes en el sistema, y enviar mensajes a otro usuario a través del comando MAILBOX. Otro de los niveles de protección son las claves o «passwords»: para que un usuario sea reconocido por el sistema y de esta forma pueda acceder a los ficheros asociados, debe conocer esta clave o «contraseña».



El individuo A tiene, en este caso, prioridad 1, y el B, prioridad 3. A puede acceder a todos los ficheros almacenados.

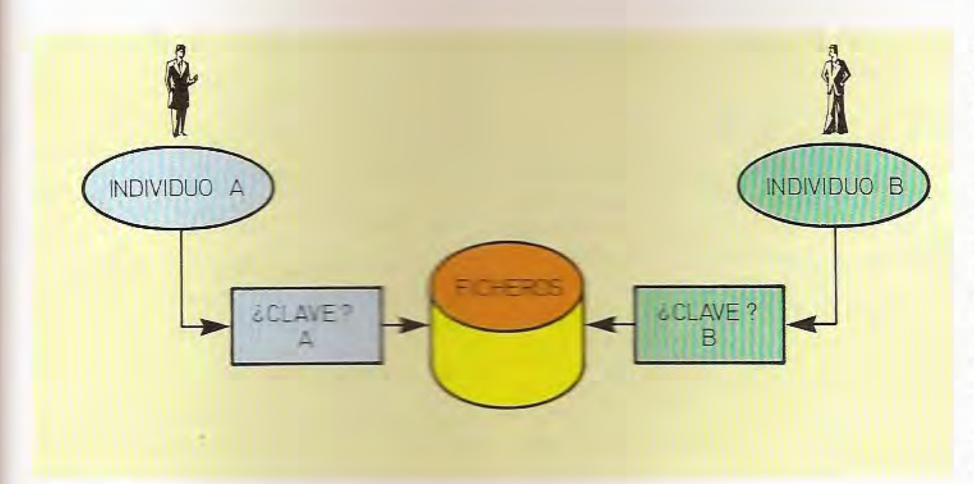


Los ficheros públicos son accesibles por todos los usuarios del sistema, independientemente de la prioridad que tenga cualquiera de ellos.

Tipos de ficheros

El OASIS soporta hasta seis tipos diferentes de ficheros:

- *ASCII:* ficheros formados por caracteres en código ASCII. Los registros asociados son de longitud variable. El acceso a cada uno de ellos es secuencial. Ello comporta un inconveniente: para acceder al registro N hay que leer



Los archivos privados tienen asignada una clave o «contraseña». De esta forma el acceso a ellos está limitado a aquellos usuarios que conozcan esta clave.



El sistema operativo OASIS está implantado en muchos microordenadores cuya unidad central de proceso es un Z-80. Aunque estos sistemas son capaces de trabajar con CP/M, que presenta la ventaja de una enorme biblioteca de software, el OASIS dispone de muchos más comandos y utilidades.

antes los N-1 anteriores. La actualización de este tipo de ficheros se efectúa añadiendo al final de cada uno de ellos los registros necesarios.

• *Directos:* estos ficheros contienen datos binarios. En el momento de su creación, operación que se realiza mediante el comando CREATE, se fija la longitud y el número de los registros. Estas características son fijas, es decir, no pueden modificarse posteriormente. El acceso a estos ficheros se realiza mediante el comando RECORD.

• *Indexados:* el modo de acceso es la principal característica de estos ficheros. Este se realiza mediante una referencia alfanumérica que permite al sistema encontrar el registro correspondiente. Las claves de cada uno de ellos están ordenados para su posterior lectura.

• *Claves:* este tipo de ficheros se organiza de forma similar a los indexados. La única diferencia estriba en la organización de las claves que en este caso no están ordenadas.

• *Absolutos:* el montador de enlaces del sistema es quien crea estos ficheros. Contienen programas en código máquina. Su información es imagen del contenido de la memoria.

• *Reubicables:* ficheros también creados por el montador de enlaces. La dirección de carga es la posición cero. Cuando uno de ellos es llamado por algún programa, el sistema lo coloca en las posiciones de memoria que están libres. Esta característica les diferencia de los ficheros absolutos, pues éstos siempre se cargan en direcciones fijas.

Protecciones del sistema

El sistema incluye diversos tipos de protecciones para los ficheros. De esta forma, mediante el comando RENAME se impide la modificación o el borrado del fichero asociado.

La ejecución de los programas también tiene niveles de protección. Si se trabaja en un entorno multiusuario, cualquier fichero completo o cualquier registro se puede cerrar de forma que el resto de los usuarios no puedan acceder a él hasta que vuelva a ser abierto.

Ayudas del sistema

Uno de los objetivos de diseño del OASIS ha sido la sencillez de su manejo. Por ello se ha dotado a cada comando de una información general de ayuda. Esta se sirve de una forma muy simple: el usuario sólo tiene que teclear HELP, un espacio y el nombre del comando. El sistema responde entonces con la información sobre el comando: lo que hace, su sintaxis y las diferentes opciones disponibles.

Controles y funciones opcionales

El sistema reconoce una serie de teclas como controles y funciones opcionales:

- *SYSTEM RESTART*

Obliga al sistema a salirse del programa en curso y efectuar un nuevo bootstrap. Tras su pulsación el ordenador pide la confirmación de esta salida.

- *SYSTEM CANCEL*

Permite romper la ejecución de un programa y pasa el control al nivel superior. Este es normalmente el sistema. Se cierran todos los ficheros y se restablecen valores de defecto.

- *PRINTER ECHO*

Funciona como un conmutador. Cuando está en ON saca por la impresora la información que se visualiza en la consola. El valor por defecto es OFF.

- *CONSOLE ECHO*

Igual que Printer Echo, funciona como un conmutador. Cuando está en ON visualiza en pantalla la información tecleada. El valor por defecto es ON.

- *PROGRAM PAUSE*

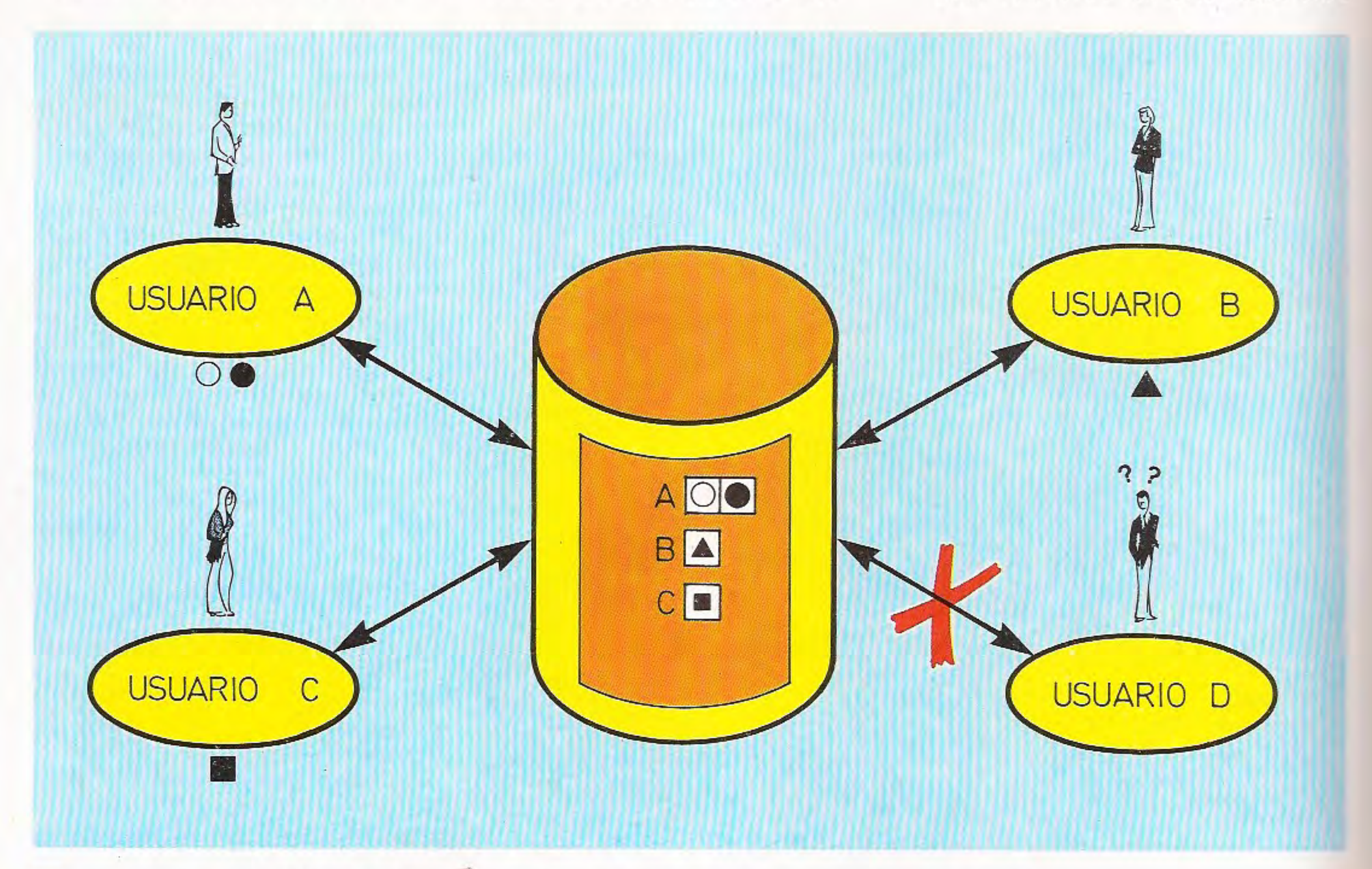
Permite interrumpir la ejecución de un programa, que continuará cuando vuelva a pulsarse este comando.

- *PROGRAM CANCEL*

Esta tecla puede estar desactivada durante la ejecución de un programa. El Basic la utiliza para volver al modo comando, DEBUG la utiliza para terminar con un comando en ejecución.

- *DEBUG BREAK*

Trae consigo la entrada en modo DEBUG. Para entrar en este modo debe



Los usuarios A, B y C tienen abierto el acceso a los archivos que pertenecen, respectivamente, a cada uno de ellos. Por el contrario, el usuario D no tiene definido un «account» o zona interna propia, de ahí que no tenga acceso a ningún archivo de información.

asegurarse de que el Debugger está cargado.

- *LINE CANCEL*

Permite ignorar la última línea introducida.

- *CHARACTER DELETE*

Borra el último carácter introducido.

- *CONSOLE DISPLAY FAST* y *CONSOLE DISPLAY SLOW*

Pasan del estado de salida rápida a salida con retardo.

- *CONSOLE SCREEN WAIT*

Este comando es del tipo interruptor. Con él conectado, el sistema espera a que se pulse la barra espaciadora o la tecla RETURN para mostrar la siguiente página en pantalla.

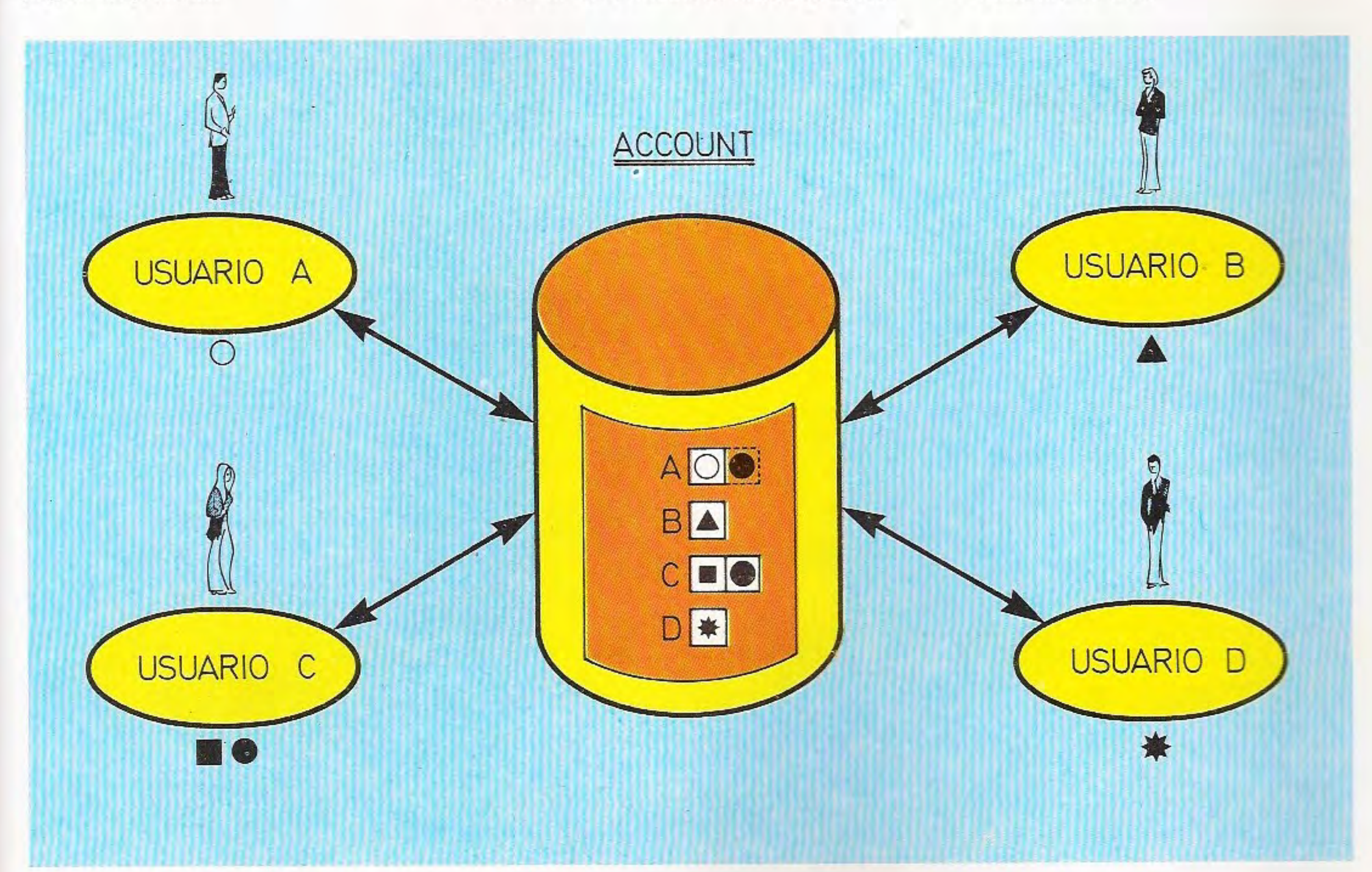
Ficheros del OASIS

Los nombres de los ficheros soportados por este sistema operativo se definen en base a un nombre y a un tipo, ambos de ocho caracteres. Estos pueden ser sustituidos en la definición por ciertos símbolos genéricos (comodines). De esta forma, el carácter «*» significa que tras esta posición pueden colocarse cuantos caracteres se deseen. El símbolo «?» acepta cualquier carácter colocado en esta posición.

Clasificación de los comandos

El filtro a través del cual entran las órdenes de los comandos antes de ejecutarse es el CSI (Command String Interpreter). El comando es una palabra inglesa, asociada a la acción a ejecutar, compuesta de un número de caracteres nunca superior a ocho. Los parámetros deben estar incluidos entre paréntesis. Los comandos pueden tener asignadas las siguientes operaciones:

- Mantenimiento del disco.
- Mantenimiento y control de ficheros.
- Desarrollo y mantenimiento de ficheros.
- Cambio de los parámetros del sistema.
- Diagnóstico del sistema.
- Ejecución de programas.
- Comunicaciones.



Para que el usuario D pueda operar con el ordenador y acceder a sus propios archivos, hay que empezar definiendo un nuevo usuario o «account» en el sistema. Esto se realiza por medio del comando ACCOUNT.

Principales comandos del OASIS

Una de las formas de caracterizar a un sistema operativo es mediante sus comandos. Entre los más destacables del OASIS se encuentran los siguientes:

- ACCOUNT

Este comando permite crear, modificar o borrar niveles de usuario. Mediante esta palabra se define el nombre del fichero, se determina qué usuarios tienen acceso a él, el «password» y el nivel de privilegio asociado. Este comando puede cambiar toda la configuración lógica exterior.

- ATTACH

Asocia programas de control de entrada/salida a dispositivos lógicos. De esta forma es posible configurar el sistema dependiendo de los periféricos disponibles. Permite definir, por ejemplo, los parámetros descriptores de una entrada/salida en serie, la longitud de línea de un listado, el protocolo de comunicación síncrona o asíncrona, el número de errores permitidos en el acceso al disco, etc.

- CHANGE

Se utiliza para cambiar el nivel de privilegio de los programas.

- FORCE

Fuerza a que un usuario pase a ejecutar una determinada tarea.

- LOGON y LOGOFF

La primera palabra permite la entrada del usuario al sistema. La segunda permite al usuario salirse del sistema.

- MSG y MAILBOX

Estos comandos sólo tienen sentido en configuraciones multiusuario. El primero manda a un usuario, o a todos los que estén conectados al sistema, un determinado mensaje desde alguno de los terminales. Si los receptores no pueden recibirlo, porque no están activados, el sistema pregunta al emisor si quiere conservarlo en el MAILBOX del destinatario. Al hacer un LOGON, el usuario receptor recibe el mensaje almacenado.

- PEEK

Especifica un usuario, de forma que los datos de su consola pasen también a la del que ejecuta el comando.

- SET

Este comando permite modificar ciertos parámetros y opciones del sistema, como, por ejemplo, la ficha.

- SHARE

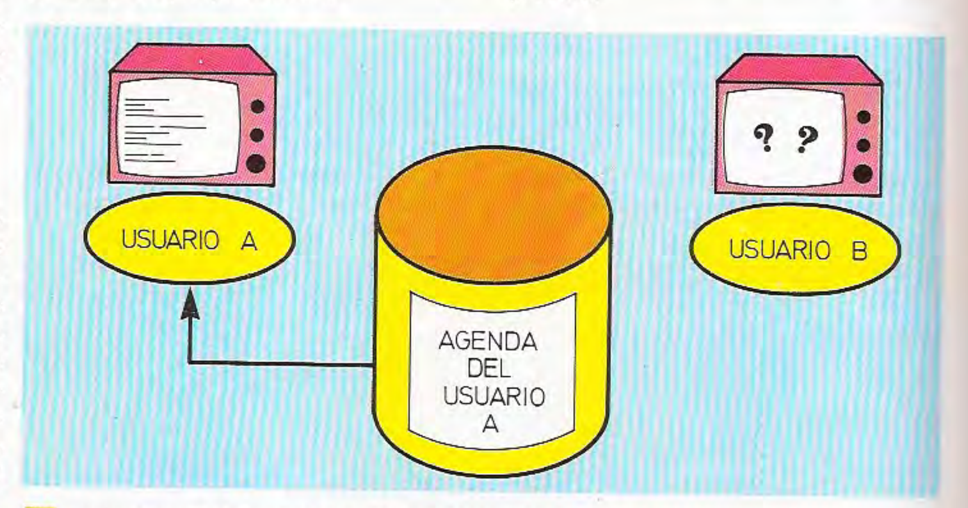
Permite que el usuario especificado acceda a un determinado fichero.

- SHOW

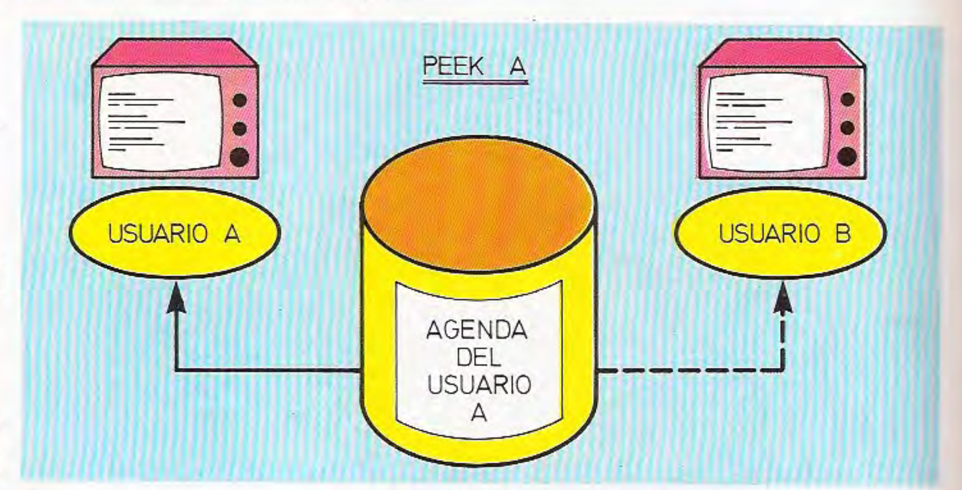
Visualiza el valor de ciertos parámetros, como, por ejemplo, el nivel de privilegio, el indicador de recepción de mensajes, el uso de memoria o los usuarios conectados al sistema.

- SPOOLER

Pasa el control al spool de la impresora.



Aún cuando la explotación de un determinado archivo es exclusiva del usuario al que pertenece, las posibilidades «multiusuario» del OASIS permiten acceder a la visualización de archivos ajenos...



...Si el usuario B ejecuta el comando PEEK A en el instante en el que el usuario A está realizando una consulta a un archivo propio (su agenda, por ejemplo), el terminal del usuario B presentará la información visualizada en la pantalla del A.

Indice temático

Dialogando con el ordenador

Los lenguajes informáticos

El lenguaje máquina

Lenguajes de «ensamble»

Lenguajes de alto nivel

Cuatro lenguajes evolucionados

Del Pascal al ADA

Cuadros

Métodos de proceso de datos

Cuadros

El sistema operativo

Cuadros

Sistemas operativos para microordenadores

Introducción al CP/M

Cuadros

Introducción al OASIS